TAMPERE UNIVERSITY OF TECHNOLOGY

ABHISHEKH GUPTA

STOCHASTIC PROCESSES AS A SOURCE OF CELL TO CELL DIVERSITY AND CELLULAR AGEING

Master of Science Thesis

Examiners: Andre S. Ribeiro
Imed Hammouda

Examiners and subject approved in the Faculty of Computing and Electrical Engineering Council meeting on 09.05.2012

ABSTRACT

TAMPERE UNIVERSITY OF TECHNOLOGY
Degree Programme in Information Technology
ABHISHEKH GUPTA: Stochastic processes as a source of cell to cell diversity and cellular ageing
Master of Science Thesis, 58 pages
June 2012
Major subject: Software Systems
Examiners: Andre S. Ribeiro, Imed Hammouda
Keywords: Gene Expression, Cell division, Partitioning, Models, Cell-to-cell diversity, Ageing, Stochastic Simulation Algorithm, Compartments

Even populations of monoclonal cells exhibit phenotypic diversity. There are several sources generating such diversity, including stochasticity in the dynamics of gene expression, and the stochastic partitioning of molecules during division. This thesis focuses on the construction and simulation of a realistic model of gene expression and on the stochastic partitioning of cellular components during cell division.

First, we present and make use of statistical methods to extract information on the kinetics of gene expression from live-cell measurements at the single RNA molecule level. This information allows us to characterize the kinetics of the multi-stepped process of transcription initiation, including the degree of noise in transcript production, as well as the kinetics of partitioning of protein aggregates by the cell's poles. A model of single gene expression in a growing population of cells and a model of ageing in bacteria are then constructed based upon these measurements.

Next, we present a new simulator which uses the Stochastic Simulation Algorithm to simulate the dynamics of intracellular processes in populations of cells, each of which able to grow and divide with random partitioning of molecules. Cells are represented in the simulator by compartments that can be created and destroyed at runtime. Logarithmic simulation algorithms and efficient data structures were designed and are here presented, which minimize the computational cost of simulating the dynamics of large cell populations that involve a large number of chemical reactions.

PREFACE

This Master's thesis was carried out between 2011-2012 at the Computational Systems Biology Research Group in the Department of Signal Processing, Tampere University of Technology, Tampere, Finland.

I would like to thank Jason Lloyd Price for his continuous support and technical assistance and Meenakshisundaram Kandhavelu for providing the experimental images for the measurements.

I would also like to extend my gratitude to Asst. Prof. Andre S. Ribeiro and Adjunct Prof. Imed Hammouda, for supervising this thesis to its completion.

Tampere, May 21st, 2012

Abhishekh Gupta

CONTENTS

TERMS AND DEFINITIONS

aTc	anhydrotetracycline
CME	Chemical Master Equation
DM	Direct Method
DNA	Deoxyribonucleic Acid
FRM	First Reaction Method
GFP	Green Fluorescent Protein
IPTG	Isopropyl-β-D-thiogalactopyranoside
KDE	Kernel Density Estimation
LDM	Logarithmic Direct method
mRNA	messenger Ribonucleic Acid
NRM	Next Reaction Method
ODM	Optimized Direct Method
PCA	Principal Component Analysis
PDM	Partial Propensity Method
RBS	Ribosome Binding Site
RNA	Ribonucleic Acid
RNAp	RNA polymerase
SDM	Sorting Direct Method
SPDM	Sorting Partial Propensity Method
SSA	Stochastic Simulation Algorithm
SSA-CR	SSA with Composition and Rejection
TSS	Transcription Start Site

1. INTRODUCTION

All biochemical processes in cells are inherently stochastic. This is of relevance to cells' behaviour, as in several of these internal cellular processes the intervenient macromolecules exist in low copy numbers. This implies that the occurrence of single events such as the production or degradation of a single molecule may have detectable effects on the dynamics of the cell. Cellular processes where stochasticity plays a major role include gene expression and responses to external stimuli. Since these processes affect, to great extent, the cell's well-being, it is of importance to understand their statistical properties [1-4] since these are responsible, among other things, for the cell-to-cell diversity of a population. For that, it is of interest to construct realistic models of these processes, and to simulate them, as they allow exploring behavioural patterns and the effects of interval variables in these processes, which would not be possible, or easily achievable, when using an experimental approach alone.

Several studies have reported sources of phenotypic diversity in monoclonal cell populations. In this thesis we focus on constructing models to study two of these sources, namely, gene expression [1,5,6] and errors in partitioning of molecules during cell division [7]. Our aim is, from observations of these processes in live cells, to develop a simulator capable of mimicking them in a realistic fashion, so that it can be used as a test bed for future experiments. To model and simulate these complex processes in a realistic way we need to consider the underlying events that compose them, namely their kinetics.

The first process, gene expression has a dynamics that is mostly regulated at its initial stage, i.e., transcription initiation, a multi-stepped process [8,9]. *In vitro* studies have measured the mean durations of some of these steps [10-12]. However, it is unknown to what extent do these results apply to living bacteria since, for example, they are not well-stirred environments. Only recently have *in vivo* methods been developed to detect *in vivo* individual RNA molecules as these are produced [13]. These rely on the tagging of target RNA molecules with fluorescent proteins.

The other process considered here, i.e. the segregation of molecules such as RNAs and proteins during cell division, is also a source of cell to cell diversity due to, at least, inevitable deviations from a perfectly even partitioning of molecules between daughter cells (also referred to as "partitioning errors") [7]. Notably, different molecules segregate into the two daughter cells in different manners, which implies that while the process of segregation is stochastic, it is likely not 'purely random'. As an example of a non-purely random partitioning scheme, a study has shown that, in *Escherichia coli*,

some unwanted protein aggregates tend to preferentially accumulate at one of the cell poles (the older pole), leading to a large difference in the amount inherited by the daughter cells [14]. This has been linked to ageing in these organisms, since daughter cells that inherit the older pole exhibit decreased vitality, i.e., rate of division [15].

The methods developed here to model these processes in dynamic populations of cells are such that they can be incorporated onto Monte Carlo simulations of the Chemical Master Equation (CME), given the success of this approach in modelling gene expression [1,16-19]. The canonical implementation of this method is the Stochastic Simulation Algorithm (SSA) [20].

The SSA in its original version [20] cannot be used to simulate the models we aim to construct. First, some events (e.g. protein folding) need to modelled as 'delayed events', that is, as non-instantaneous, since they do not consist of bimolecular events. Since all we aim is to model the kinetics of the processes, not the physical processes, e.g. this problem is overcome by using algorithms such as the delayed stochastic simulation algorithm [21], as these allow some events to not be instantaneous. Additionally, a dynamic population of cells requires a dynamic number of reactions since one cannot predict, beforehand, what molecules will be present in the system at any given time and thus what reactions need to be modelled. The concept of dynamic compartments is therefore needed, as they allow sets of reactions to be introduced and removed at runtime, depending on the evolution of the system [22].

Other problems also require addressing. For example, during the division of a cell, all molecules within it must be partitioned between the daughter cells, according to some specified segregation mechanism. Including such component to existing stochastic simulators (e.g. SGNSim [23]) would allow, e.g., exploring *in silico* the effects of partitioning in the diversity of cell populations [18].

This thesis aims to construct a simulator able to mimic the temporal evolution of cell-to-cell diversity in populations due to noise in gene expression, and stochastic partitioning schemes in division. For that, first, we measure the kinetics of transcription initiation in live bacteria in order to obtain experimental data (section 5.1) that will be used to develop a more realistic model of single gene expression (section 5.3). We also present measurements relative to the partitioning of unwanted protein aggregates (section 5.2). To extract the information from the measurements, new methods were developed that are also presented here. Namely, we describe the methods developed to measure distributions of intervals between consecutive transcription events from measurements in *E. coli* cells [24,25] (section 3.1), and to measure biases in the spatial segregation of unwanted aggregates by these cells' poles (section 3.2).

Finally, we present the simulator of dynamic cell populations capable of modelling stochastic, compartmentalized processes inside cells, and probabilistic partitioning of molecules upon cell division (section 4). The simulator is built using a combination of efficient simulation algorithms so that large chemical systems and large populations of cells may be simulated in reasonable time.

2. BACKGROUND

In this section, we first present the biological processes that we focus on. It is followed by the algorithms and methods currently used to simulate these processes.

2.1. Stochastic Processes in Bacteria

Gene Expression is the process by which information encoded in a gene's DNA sequence is read to synthesize, first, an RNA molecule and, from this RNA, a protein. Stochastic events in this process have been found to be responsible for fluctuating time patterns of RNA and protein numbers in individual cells [1]. These fluctuations are a source of diversity of protein concentrations between cells at any given time, and thus of phenotypic diversity. For example, stochastic variations in the concentrations of two independent regulatory proteins competitively controlling a switch point in a pathway can lead to probabilistic pathway selection. One consequence of this is that an initially homogeneous cell population may partition into distinct phenotypic subpopulations [1].

Bacteria have been used as model organism to study stochasticity in gene expression. In these organisms most Ribonucleic acid (RNA) molecules exist in very small numbers. In particular, for most RNAs, only one to a few molecules is observed at any given moment in a cell [26]. It has also been shown that the phenotype of these cells is affected by the number of RNA molecules produced by each gene [27] and the timing with which they are produced, because protein numbers follow the RNA numbers [19,28].

Transcription is the process by which genetic information stored in a DNA strand is copied into a complementary strand of RNA, with the aid of RNA polymerases. One transcription event results in an RNA copy of a gene, which generally codes for a functional protein. Protein coding RNAs are called messenger RNA (mRNA). Transcription begins when the RNA polymerase binds to the promoter region of the gene. Once bound, the RNA polymerase unwinds a small section of the DNA, after which elongation begins where it uses one strand, known as the coding strand, as a template from which to synthesize an exact RNA copy [11]. In prokaryotes like bacteria, transcription is coupled with translation, which is the process of protein synthesis from a specific sequence of amino acids. This is accomplished by a protein/RNA hybrid known as ribosome. Ribosomes bind to the translation initiation sequence on the mRNA, and elongate the protein in a similar manner to transcription, creating a new protein.

In prokaryotes, both transcription and translation are stochastic, multi-stepped processes. The events in transcription and in translation are probabilistic in nature

[18,19,29,30], and their kinetics is sequence dependent [5,9,31,32]. In bacteria, stochastic events that occur during transcription initiation [1,6], elongation [6], and mRNA degradation [32], among others, cause fluctuations in the mRNA count [13,33-35].The production of mRNAs, one at a time, in live cells thus needs to be observed to understand the dynamics of transcription, including initiation and elongation.

2.1.1. Initiation

Transcription in a bacteria starts when an RNA polymerase (RNAp) diffusing along the DNA strand binds to a promoter region [36]. This process is referred to as the closed complex formation. Before a productive elongation can occur, the DNA must be bent and unwound (the open complex formation, during which the RNAp places itself in the transcription start site (TSS)), and the RNAp's clamp/jaw must assemble on downstream DNA [37]. Initiation is thus a complex process consisting of a series of events which take a non-negligible amount of time to be completed once started [9]. Once these events are completed, the RNAp begins elongating the complementary RNA strand and clears the promoter, allowing another RNAp to bind.

Several studies on the kinetics of transcription initiation have measured the mean durations of some of these steps using *in vitro* measurements [9-12,38]. The results suggest that the two most rate-limiting steps in initiation are the closed and the open complex formations [9]. The durations of these steps vary widely between promoters [5], as well as with temperature [36] and concentration of possible activator and repressor molecules [9].

It has been suggested that the dynamics of gene expression, and therefore one of the key determinants of cellular phenotype, is largely controlled at the level of initiation [19,28,39]. The amount of variability in this process is therefore one of the major contributors to the phenotypic diversity in a monoclonal bacterial population.

2.1.2. Elongation

Translation in bacteria can begin before the transcription is completed, and several translation events can occur in parallel from one transcript. However, since translation cannot and produce functional proteins before transcription completes, events during transcription elongation can affect the mean and fluctuations of protein levels. Studies have shown how events during transcription elongation can affect translation elongation and thereby produce fluctuations in protein levels [40].

A delayed stochastic model of transcription and translation including events such as the promoter open complex formation and alternative pathways to elongation, namely pausing, arrests, editing, RNA polymerase traffic, and premature termination was proposed and used to investigate these effects [40]. Although in some cases, such as certain exceptional sequence-dependent pauses [5], these events may cause non-negligible fluctuations in RNA and protein numbers, since elongation does not take a large amount of

time relative to other processes in gene expression [41], these events are not expected to significantly affect gene expression dynamics for most genes [25].

2.1.3. Degradation

In all living cells, RNA degradation is essential to control the steady-state concentration of mRNA. It has been found that the enzymes responsible for RNA degradation and processing are able to assemble into a large multi-protein complex, the RNA degradosome that consists of RNase E and other enzymes [42]. In this complex, RNase E has the endoribonucleolytic activities that are responsible for the degradation of mRNAs. The rate at which this is performed determines the time to reach steady state and thus affects the gene's reaction time to external stimuli. Intuitively, it must be fast to responding to external signals and slow for normal regulation processes.

In bacteria, mRNA degradation is often modelled as a first-order reaction. The half-life of an mRNA typically ranges from 3 to 8 minutes [28]. No general relationship was found between an mRNA's abundance and that mRNA's half-life, indicating that though mRNA decay is an important part of the cellular system, it is not generally used by cells as a regulatory mechanism of RNA and protein abundances [28]. The added cell-to-cell diversity introduced by such a first-order decay process is known [29].

2.1.4. Partitioning and Ageing

There are other, non-genetic sources of phenotypic diversity between the daughter cells of a bacterial cell. Because of the low copy number of some molecules in the cells, imperfections in a purely unbiased partitioning will lead to increased phenotypic diversity in a monoclonal population [7,43]. For example, if a particular molecule species segregates independently and randomly into either daughter, this will result in a Binomial partitioning distribution upon division, which will double the cell-to-cell diversity after the division, in terms of the normalized variance of the molecule numbers in the population. If the molecules form dimers which split evenly into the daughter cells upon division, this will lead to a partitioning distribution with less variance than Binomial. Consequently, the partitioning of the molecules at division will introduce less diversity into the population, and is thus more "ordered" than independent Binomial partitioning. In contrast, if the molecules form clusters, and the clusters segregate independently, this will lead to a partitioning distribution with greater variance than Binomial [7]. This "disordered" partitioning then leads to considerably greater cell-to-cell diversity in the population after division.

Of particular interest are partitioning schemes in which there is a purposeful asymmetry in division, even in an organism with apparently morphologically symmetric division, such as *E. coli*. In this case, molecules preferentially segregate to one of the poles prior to division, and thus preferentially partition into one of the daughter cells [44]. This biased Binomial partitioning is a disordered partitioning scheme which en-

hances cell-to-cell diversity of RNA and protein numbers [44]. Such a partitioning scheme has been observed in *E. coli*, whereby unwanted protein aggregates tend to accumulate at the older pole of the mother cell [14]. This accumulation has been linked to decreased vitality, i.e. a slower rate of division, in the daughter cell inheriting the older pole [14]. This suggests that the two daughters of an *E. coli* cell should not be considered as sister cells, but rather as an ageing parent repeatedly giving birth to rejuvenated offspring [15].

2.2. Chemical Master Equation

Models of the stochastic biochemical processes in bacterial cells must account for the above effects. That is, they must accurately capture the stochastic nature of the events occurring in the system. The stochastic formulation of chemical kinetics [45] has successfully been applied to these models. This formulation describes the time-evolution of a well-stirred set of chemically interacting molecules in thermal equilibrium in a fixed reaction volume. The state of a system with N chemical species at time t is represented by an N-dimensional vector $\mathbf{x}$ containing the number of molecules of each species in the volume. The change in the population of a species is a consequence of the occurrence of one of M chemical reactions that can take place between the molecules. The time-evolution of $\mathbf{x}$ takes the form of a random walk through the N-dimensional space of the populations of the reacting species. This approach to the kinetics of a spatially homogenous system of reacting chemicals is based on the probabilities of occurrence per unit time of each of these reactions R_μ, based on the current state vector, which are defined by the propensity function a_μ:

$$a_\mu(\mathbf{x})dt \equiv \text{the probability that a particular combination of the molecules that are presently in the system will react via reaction } R_\mu \text{ in the next infinitesimal time interval } [t, t + dt). \quad (1)$$

This definition on its own can be used to derive the master equation for a chemical system via the laws of probability. Its existence can thus be considered to be the fundamental premise of the stochastic formulation of chemical kinetics. The form that the function a_μ takes depends on the type of the reaction it represents. The physical rationale for unimolecular and bimolecular reactions are as follows.

Unimolecular reactions occur internally within each molecule of a given species S_i and are usually quantum mechanical in nature. The underlying physics dictates that there is some constant c_μ such that $c_\mu dt$ gives the probability that some molecule of S_i will spontaneously react via reaction R_μ in the next infinitesimal time dt [46]. From the laws of probability, it follows that if there are X_i molecules of S_i in the system at a given instance, the probability that one of them will react via R_μ in the next infinitesimal time interval is $X_i c_\mu dt$. Thus, the propensity function for unimolecular reactions is $a_\mu(\mathbf{x}) = X_i c_\mu$.

Bimolecular reactions involving two different molecular species S_i and S_j occur when two such molecules meet and react. Using the homogeneity assumption, it can be shown that there exists a constant c_μ such that $c_\mu dt$ gives the probability that a given pair of molecules will meet and react via reaction R_μ [47]. This constant can be derived from microphysical properties [48]. If there are X_i molecules of S_i and X_j molecules of S_j currently in the system, then there are $X_i X_j$ pairs of these molecules. The probability that one of these pairs will meet and react via R_μ in the next infinitesimal time dt is therefore $X_i X_j c_\mu dt$. The propensity function for bimolecular reactions with different chemical species is then $a_\mu(\mathbf{x}) = X_i X_j c_\mu$.

For bimolecular reactions between two molecules of the same species S_i, the number of pairs does not grow as X_i^2, since a molecule cannot react with itself and the pairs $X_i - X_j$ and $X_j - X_i$ must be counted only once. Instead, the number of pairs grows as $X_i(X_i - 1)/2$, making the propensity function for such a reaction $a_\mu(\mathbf{x}) = X_i(X_i - 1)c_\mu / 2$.

In the stochastic formulation, we would like to describe the joint probability distribution $P(\mathbf{x}, t \mid \mathbf{x}_0, t_0)$ of having a given state vector $\mathbf{x}$ at time t after the initial conditions $\mathbf{x} = \mathbf{x}_0$ at $t = t_0$. Also, we assume that v_i represents the absolute number of each reactant that change when reaction R_i occurs. Using equation (1), the rate of change of the probability of being in a given state $\mathbf{x}$ can be expressed as the sum of the probabilities of all reactions that can change the system's state into $\mathbf{x}$ in the next infinitesimal time interval, subtracting the sum of the probabilities of all reactions that can cause the system to leave that state. The result is a partial differential equation for P, the Chemical Master Equation (CME) [45]:

$$\frac{\partial P(\mathbf{x}, t \mid \mathbf{x}_0, t_0)}{\partial t} = \sum_{\mu=1}^{M} \left[a_\mu(\mathbf{x} - \nu_\mu) P(\mathbf{x} - \nu_\mu, t \mid \mathbf{x}_0, t_0) - a_\mu(\mathbf{x}) P(\mathbf{x}, t \mid \mathbf{x}_0, t_0) \right] \tag{2}$$

This equation determines the probability that each species will have a specified molecular population at a given future time. The function that satisfies the CME simultaneously describes the probability of all possible trajectories through the N-dimensional state space of reactant populations. Because of the explicit handling of every possible state that the system can be in, the CME can take an accurate account of the effects of both fluctuations. This has been a major justification for using the stochastic approach over the mathematically simpler deterministic approach.

2.3. Stochastic Simulation Algorithm

As the CME in equation (2) can rarely be solved analytically, simulated trajectories of $\mathbf{x}$ versus t are often used to sample the distribution of $\mathbf{x}$ instead. This is not the same as solving the CME numerically, as that would give us the probability density function of $\mathbf{x}$ instead of a random sample of $\mathbf{x}$. The key to generating simulated trajectories of $\mathbf{x}$ is

not the function $P(\mathbf{x},t \mid \mathbf{x}_0,t_0)$, but rather a new probability function $p(\tau,\mu \mid \mathbf{x},t)$ which is defined as follows:

$p(\tau,\mu \mid \mathbf{x},t)d\tau$ = the probability that the next reaction in the system will occur in the infinitesimal time interval [$t + \tau$, $t + \tau + d\tau$), and will be an R_μ reaction. (3)

If the system is currently in state **x**, this function is the joint probability density function of the two random variables: the time until the next reaction occurs (τ) and the index of this reaction (μ). By applying the laws of probability, an exact formula for $p(\tau,\mu \mid \mathbf{x},t)$ is derived from the fundamental premise equation (1) [47].

$$p(\tau,\mu \mid \mathbf{x},t) = a_\mu(\mathbf{x})\exp(-a_0(\mathbf{x})\tau \quad (4)$$

where,

$$a_0(\mathbf{x}) = \sum_{\mu=1}^{M} a_\mu(\mathbf{x}).$$

This equation is the mathematical basis for the stochastic simulation approach. One implementation of this approach is Stochastic Simulation Algorithm (SSA), a Monte Carlo procedure for numerically generating time trajectories of the molecular populations in exact accordance with the CME. The procedure that SSA follows can be listed as [20]:

1. Set time $t = 0$. Set up the initial state vector $\mathbf{x} = \mathbf{x}_0$.
2. With the system in state **x** at time t, evaluate all the $a_\mu(\mathbf{x})$ and their sum $a_0(\mathbf{x})$.
3. Using a suitable sampling procedure, generate a random pair (τ,μ) according to the joint probability distribution in equation (3)
4. Output the system state for each sampling point in the time interval $[t, t + \tau)$.
5. If $t + \tau \geq t$ stop, terminate.
6. Set $t = t + \tau$, and $\mathbf{x} = \mathbf{x} + \nu_\mu$.
7. Recalculate a_i for all i such that any X_μ that was changed in step 4 appears as a reactant in R_i.
8. Go to step 3.

There are several exact procedures for generating samples of τ and μ according to the joint probability distribution as mentioned in step 3. The two original, statistically equivalent sampling procedures of the SSA [20] are the Direct Method (DM) and the First Reaction Method (FRM). The other methods presented here, specifically the Next Reaction Method (NRM), the Logarithmic Direct method (LDM), and the Partial Propensity Method (PDM), are all based (in one way or another) on either DM or FRM.

2.3.1. Direct Method

The Direct Method applies the standard inversion generating method of Monte Carlo theory [20]. In this sampling approach, two random numbers r_1 and r_2 are drawn from the uniform distribution in the unit interval, and the random pair (τ,μ) is computed as:

$$\tau = (1/a_0(\mathbf{x}))\ln(1/r_1) \tag{5}$$

$$\mu = \text{the smallest integer satisfying } \sum_{\mu'=1}^{\mu} a_{\mu'}(\mathbf{x}) > r_2 a_0(\mathbf{x}). \tag{6}$$

The formulae used in this method can be derived on the basis that any two-variable probability density function can be written as the product of two one-variable probability density functions. That is, if $P_1(\tau)\mathrm{d}\tau$ is the probability that the next reaction will occur between times $t+\tau$ and $t+\tau+d\tau$, irrespective of which reaction it might be; and $P_2(\mu\,|\,\tau)$ is the probability that the next reaction will be an R_μ reaction, given that the next reaction occurs in $[t+\tau, t+\tau+d\tau)$, the probability density function in equation (3) can be written as:

$$P(\tau\,|\,\mu) = P_1(\tau)\cdot P_2(\mu\,|\,\tau)$$

This equation can be rewritten in the form:

$$P_1(\tau) = \sum_{v=1}^{M} P(\tau\,|\,v)$$

and,

$$P_2(\mu\,|\,\tau) = \frac{P(\tau\,|\,\mu)}{\sum_{v=1}^{M} P(\tau\,|\,v)}$$

Using the laws of probability, these have been shown to yield [20]:

$$P_1(\tau) = a_\mu \exp(-a_0\tau) \tag{7}$$

$$P_2(\mu\,|\,\tau) = a_\mu / a_0 \tag{8}$$

The core idea of the direct method is to first generate τ according to $P_1(\tau)$ and then generate μ according to $P_2(\mu\,|\,\tau)$. Since P_2 is independent of τ, this can be accomplished with the formulae in equation (5). The resulting random pair (τ, μ) will be distributed according to $P(\tau\,|\,\mu)$. The Direct Method, given a fast, reliable uniform random number generator, is easily programmable. This method is therefore a simple, rigorous procedure for implementing Step 3 of the SSA.

2.3.2. First Reaction Method

Another sampling approach described in the original formulation of the SSA [20] is the First Reaction Method. This method, although not as efficient as the direct method, provides mathematical insights into the stochastic simulation approach. It is based on the generation of a "tentative reaction times", τ_v using a random number (r_1) from the uniform distribution in the unit interval, according to the probability density function in equation (4). This process is repeated for all reactions to compute all the respective reaction times given as:

$$\tau_v = (1/a_\mu(\mathbf{x}))\ln(1/r_1) \tag{9}$$

The pair (τ, μ) is then chosen from the M tentative reaction times such that τ is the smallest τ_v and μ is the corresponding value of v for that particular τ_v. In [20], it was

proven that the probability density function for this random pair (τ, μ) is statistically equivalent to the $P(\tau \mid \mu)$ prescribed by the CME.

Although this method is as rigorous and exact as the Direct Method, it is generally slower to compute since it requires M separate random numbers from the uniform random number generator for each of the M reactions. However, since its original publication, this method has been optimized for computational efficiency by means that do not affect its statistics.

2.3.3. Next Reaction Method

The Next Reaction Method (NRM) [49] is an approach that reduces the computational costs of the FRM. The FRM must perform O(M) operations per iteration of the SSA, since it takes a time proportional to M to both update $a_i s$, as well as to generate and identify the smallest τ_v. Instead, the NRM stores the tentative times generated in previous iterations (τ_v) in a special data structure, an Indexed Priority Queue, and reuses them where appropriate, improving its performance significantly.

The NRM uses a directed graph G(V,E) to represent the set of all reactions $\{R_v\}$ and their relationship in terms of how they affect the molecules which their propensity functions depend on [49]. In this "dependency graph", the Vertex set V = $\{R_v\}$ and there is a directed edge from v_i to v_j when reaction R_i changes a molecule's number, which affects a_j. In other words, the dependency graph is a data structure that tells precisely which $a_i s$ to change after the execution of a reaction. The use of the dependency graph limits the number calculations of a_i s to a minimum in the NRM. Since the number of edges from a given vertex in a dependency graph is typically small, only a few propensities are to be updated at each time step.

The nodes to be updated are changed in the place where they are stored, which results in the bubbling up or down the tree structure. This continues until the property of priority queue is re-established. This approach takes O(log *r*) time, where r is number of reactions in queue. But, if there are a small number of reactions, *r′*, that have rate constants faster than the others, then most updates will occur in those reactions and take approximately O(log *r′*) time. This algorithm does not continue further once it reaches a node that is already in the desired location.

This procedure requires a data structure that can efficiently handle updates to subsets of the τ_i's, and can quickly determine which is the smallest τ_i. The NRM uses an indexed priority queue to accomplish both of these goals. It can be implemented efficiently with a tree of ordered pairs of the form (i, τ_i), where i is the number of a reaction and τ_i is the putative time when reaction i occurs; and an index structure whose i^{th} element is a pointer to the position in the tree that contains (i, τ_i). Also, the hierarchy of this tree maintains the heap property in that the parents always have a lower value of τ_i than either of their children. Consequently, identifying the reaction with smallest τ_μ can then be done by examining only the root node, which makes the selection of a reaction

to run in constant time. An example of the indexed priority queue partway through a simulation is shown in *Figure 1*.

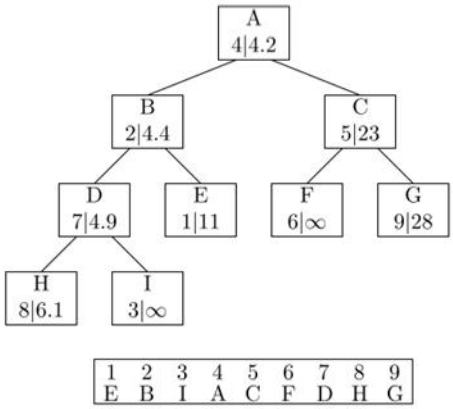

Figure 1: *Priority Queue that holds the reactions and their respective occurrence time with an indexed structure.*

The steps involved in NRM algorithm, as described in [49], are:

1. Initialize:
    a. Set initial numbers of molecules, set t= 0, generate a dependency graph G;
    b. Calculate the propensity function, a_i, for all i;
    c. For each i, generate a putative time, τ_i, according to an exponential distribution with parameter a_i;
    d. Store the τ_i values in an indexed priority queue P.
2. Let μ be the reaction whose putative time, τ_μ , stored in P, is minimum.
3. Let τ be τ_μ.
4. Change the number of molecules to reflect execution of reaction μ. Set $t \leftarrow \tau$.
5. For each edge (μ, α) in the dependency graph G,
    a. Update a_α;
    b. If $\alpha \neq \mu$, set $\tau_a \leftarrow \left(a_{\alpha,old} / a_{\alpha,new}\right)\left(\tau_a - t\right) + t$;
    c. If $\alpha = \mu$, generate a random number, ρ, according to an exponential distribution with parameter a_μ, and set $\tau_a \leftarrow \rho + t$;
    d. Replace the old τ_a value in P with the new value.
6. Go to Step 2.

This algorithm is exact and is efficient in running time, as well as in the numbers of random numbers generated in the process (only one per iteration). The run time of the NRM algorithm is not necessarily proportional to the number of reactions (theoretically it takes O(log(M))) and the total number of random numbers generated by this Method equals the sum of the number of reactions with the number of simulation events.

The NRM is thus significantly faster than both the FRM and the DM for large reaction networks containing many species and loosely coupled reaction channels. Also, the NRM requires O(log(M)) time to find the index μ of the next reaction whereas DM re-

quires O(M) time. For small systems, this advantage may not be significant since the computational cost of maintaining the complex data structures in the NRM dominates the simulation time. In addition to decreasing the complexity of the NRM, decreasing the complexity of the data structure required is the other main effort in the attempts to improve and optimize the DM for large systems.

2.3.4. Logarithmic Direct Method

The Optimized Direct Method (ODM) is an improvement of the DM [50], whereby a_0 is stored along with the other propensities. When propensity a_i is recalculated during the update, the old value is first subtracted from a_0 and the new value is added, keeping it up to date. As a result, a_0 does not need to be recalculated every reaction step, eliminating an O(M) operation from the selection step. However, the selection of index μ of the next reaction is still an O(M) operation. To minimize the impact of this linear search, Cao and colleagues [50] proposed to sort the reactions in order of decreasing a_i, allowing the linear search to terminate at an earlier i. Since the propensities can change frequently during a simulation, it was also proposed to set the reaction order by running the simulation for a short period beforehand, and sorting the reactions in descending order of the number of times each reaction occurred. Also, this method uses the concept of dependency graph from NRM, which facilitates the update of propensities of affected reactions.

The ODM was further improved by McCollum and colleagues [51] by reordering the reactions at runtime instead of having the pre-run step. Every time a reaction occurs, it is swapped with the one that is below it in the list. The set of high-frequency reactions will eventually ‘bubble up’ to the front of the reaction list. This so-called Sorting Direct Method (SDM) allows the simulator to adapt to changes in the frequencies of the reactions.

Despite their improvements, both the ODM and the SDM encounter issues in specific conditions. First, regardless of the reshuffling of reactions, the linear search remains, imposing a worst-case runtime of O(M) on the Selection step. Second, both methods suffer from a precision problem. In a digital computer, there are a limited number of bits that can be carried in the mantissa of a_0. If one reaction has a significantly higher propensity than another, then both the ODM and SDM will place it closer to the front of the list. The number of significant bits available to store the difference between the sums of a_i and a_0 may not be enough to account for the difference between them. Additionally, in simulations where the value of a_0 covers a wide dynamic range over time due to rare reactions with high propensity, a large amount of the bits in the mantissa of a_0 will be lost after the reaction with high propensity has been performed since the initial addition of its propensity will drop these bits.

The Logarithmic Direct Method (LDM) [52] is an approach that has been taken to optimize the DM avoiding such issues. In the LDM, the reaction propensities are stored in the leaves of a binary tree and the non-leaf nodes store the sum of the propensities of

their two children. Each non-leaf node thus contains the portion of the sum of a_i's which are below it in the tree and are therefore referred to as partial sums. For simplicity, this can be implemented as a single array, rather than a linked tree with special treatment for the leaves. In this case, indices below M correspond to partial sums, and indices above or equal to M correspond to propensities. Since this system stores M reaction propensities and (M−1) partial sums, it still uses Θ(M) storage space.

During the Selection step, τ can be calculated from a_0, which is readily available at the root of the tree of partial sums. μ is selected by randomizing $a = r_1 a_0$, and then performing a binary search through the tree of partial sums. It follows that the runtime of the selection step is therefore proportional to the height of the tree, Θ(logM).

During the update step, when a propensity a_i is updated, then all logM partial sums above it in the tree are also updated. This implicitly updates a_0, which can then immediately be used in the next reaction step. If R propensities are updated, then it follows that the runtime of the update step is Θ(R logM). In loosely coupled reaction systems, R is expected to vanish with M, and the runtime of this step will then be Θ(logM).

2.3.5. Partial-propensity Direct Method

The Partial-propensity Direct Method (PDM) [53] is another SSA formulation where the computational cost scales, at most, linearly with the number of species, making it appropriate for strongly coupled networks where the number of species grows slowly in comparison to the number of reactions. This computational cost is obtained by restricting the class of systems to networks containing only elementary chemical reactions, where each reaction consists of at most two reactants. If one of the species from every reaction propensity is factored out, this leads to "partial" propensities that depend on the population of at most one species. It has been shown that any non-elementary reaction can be broken down into elementary reactions, at the expense of an increase in system size [48,54]. The partial propensity of a reaction, $\pi_\mu^{(i)}$, with respect to one of its reactants is defined as the propensity of R_μ per molecule of S_i. The partial propensities of three elementary reactions are given as [53]:

1. Bimolecular reactions ($S_i + S_j \rightarrow$ Products):
   $a_\mu(\mathbf{x}) = X_i X_j c_\mu$ and $\pi_\mu^{(i)} = X_j c_\mu$, $\pi_\mu^{(j)} = X_i c_\mu$, where a_μ is the propensity of a reaction μ, X_i and X_j is the number of species S_i and S_j.
2. Unimolecular reactions ($S_i \rightarrow$ Products): $a_\mu = X_i c_\mu$ and $\pi_\mu^{(i)} = c_\mu$.
3. Source reactions ($\emptyset \rightarrow$ Products): $a_\mu = c_\mu$ and $\pi_\mu^{(0)} = c_\mu$.

The PDM uses these partial propensities and groups them such that sampling the index of the next reaction and, updating the partial propensities after a reaction has fired is performed efficiently. For the sampling step, the partial propensities are grouped according to the index of the factored-out reactant, yielding at most N + 1 groups of size O(N).

First, the index of the group is randomly selected according to the total partial propensity in the group multiplied by the group's species population, after which the partial propensity inside that group is sampled. This grouping scheme reduces the number of operations needed to sample the next reaction using a concept that is reminiscent of two-dimensional cell lists [55].

Once the selected reaction is executed, the dependency graph is used over species rather than reactions, to find all partial propensities to be updated. This is possible because partial propensities depend on the population of at most one species. Consequently the number of updates is limited to be O(N). Also, as the partial propensities of unimolecular reactions are constant and never need to be updated. In strongly coupled reaction networks, where the number of reactions dominates, this removes the dependency on the number of reactions and instead scales with the number of species. In weakly coupled networks, the scaling of the computational cost of the update becomes equal to that of methods that use dependency graphs over reactions, such as ODM, and SDM.

The Sorting Partial-propensity Direct Method (SPDM), a sorting variant of PDM, uses concepts from the SDM [51] to dynamically rearrange reactions, which in turn reduces the average search depth for sampling the next reaction in a multi-scale network. In this approach, the group index I and the element index J are bubbled up whenever reaction $\mu = L_{I,J}$ is fired. After reordering, an array is used to store I, and an array of arrays of the size of π is used for the J's. This requires additional N + M memory, but reduces the search depth to sample the next reaction. Hence, the use of partial propensities leads to SSA formulations with a computational cost that scales as some function of the number of species, rather than the number of reactions.

2.3.6. Constant-Time Direct Method

Another computationally efficient approach is an implementation of the DM based on composition-rejection sampling (CR), referred to as SSA-CR [56]. This algorithm runs in expected constant-time for both (τ, μ) generation and propensity update. This means that the computational cost of a single iteration of the SSA is neither dependent on the number of reactions nor on the number of species as in the previous methods.

The idea of rejection in the SSA-CR is illustrated in the left panel of *Figure 2*. The M reaction channels with non-zero propensity are listed on the x-axis while their respective propensities are represented by the bars with different heights. The M vertical bars are then bounded by a rectangle with height a_{max}. First, τ is chosen randomly using equation (5) with $a_0 = Ma_{max}$. Second, a random integer r_1 is the chosen between 1 and M, and a second random number r_2 is chosen in the range [0, 1). If $a_\mu < r_2 a_{max}$, reaction μ is executed, while if $a_\mu \geq r_2 a_{max}$, no reaction is performed (the sample is "rejected"). In either case, the simulation time is advanced by τ. For example, point A in the figure is rejected whereas reaction 6 is selected for point B. Thus, in each iteration, this algorithm always uses three random numbers. It is clear that the cost of executing one loop

does not depend on M, however, many of the loops will now do nothing. In order to achieve an expected constant runtime per iteration, the second part of the algorithm, composition, arranges the reactions to bind the probability of rejection.

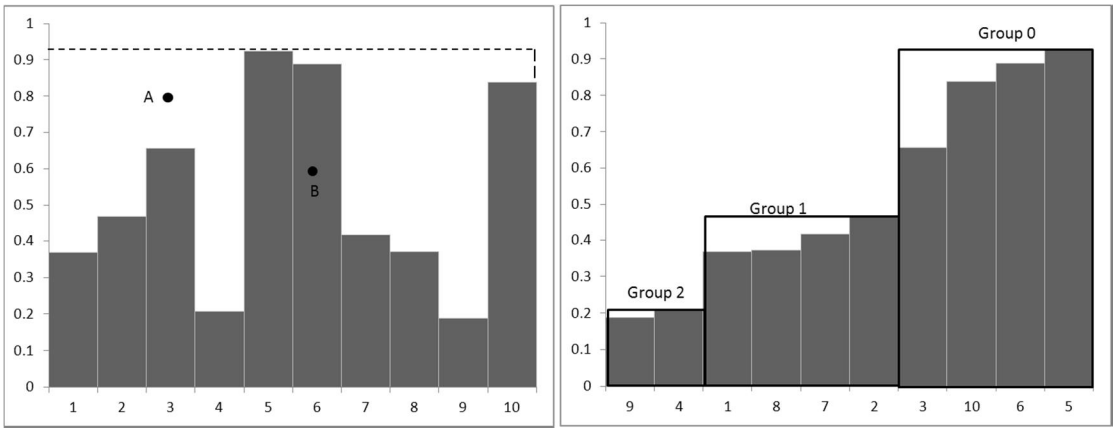

Figure 2: *Figure (left) showing the propensities of reactions bounded by a single rectangle (the rejection procedure of SSA-CR) which can be improved significantly by grouping the reaction propensities into different groups and considering separate bounding rectangle, as in figure (right) (composition procedure of SSA-CR).*

As illustrated in the right panel of *Figure 2*, the M reactions can be grouped by their propensity values into a set of groups G, such that group *g* contains reactions which have propensity between $(2^{-g-1}a_{max}, 2^{-g}a_{max}]$. The algorithm for the selection of a reaction is then composed of two stages. The first stage selects a group from G, using the area of the group's rectangle as a propensity. If the total propensity of all reactions and the rejection area in a group is p_g then the total propensity of all reactions is given as:

$$p_s = \sum_{g=1}^{G} p_g \tag{10}$$

The selection of a particular group requires one random number and a linear scan or a binary search of the G values. Once a group is selected, the second stage of the algorithm is to select a reaction within the chosen group via the rejection procedure, using a rectangle that bounds only the reactions in that group.

The number of groups required to cover the entire reaction propensity range will depend only on the ratio between the reaction with the highest propensity and the reaction with smallest non-zero propensity. In real applications, this ratio is unlikely to be dependent on N or M, since real reaction systems comprise of very large number of loosely coupled reactions. The number of groups is therefore independent of N or M, and the composition portion of the algorithm therefore runs in O(1).

In order to keep the propensities up-to-date, every time a reaction is performed, the propensity of every dependent reaction is computed and then compared to its old value. If the reaction stays in the same group, no sums must be updated. If the reaction changes group, the reaction must be removed from the old group and added to the new group, so the p_g values for both groups must be updated, as well as p_s. These are also constant-

time operations per updated reaction. For addition, a new index is added to the end of the group list, the group size is incremented, and p_g increases by $2^{-g}a_{max}$. For deletion, the reaction at the end of the group list replaces the deleted reaction, the group size is decremented, and p_g decreases by $2^{-g}a_{max}$. If the reaction propensity becomes zero, the reaction is removed from the system.

The overall SSA-CR method exhibits a desired constant-time scaling behaviour. Its scaling is O(1), independent of M and N, if the number of groups G is independent of M and N. It is therefore expected to provide a better performance than other methods for systems with a very large number of loosely coupled reactions M.

2.3.7. Delayed Stochastic Simulation Algorithm

A variant of the SSA was proposed in [49] to simulate complex processes that take a non-negligible amount of time to complete once initiated. These processes range from multi stepped processes composed of many simple reactions, such as stepwise elongation, to conformational changes in large structures, such as the unwinding of the DNA, among others. The duration of these processes within a cell can, in some case, be of an order of magnitude comparable to that of a cell's lifetime. For example, the elongation of an RNA molecule by an RNA polymerase can take as long as a few minutes in *E. coli*, whereas these cells lifetime is of the order of tenths of minutes. These processes can be represented as single-step delayed reactions, in that some of its products are released some time later than the depletion of the substrates, and not necessarily all at the same time. The disadvantage of this method is that the system's evolution in time is no longer a purely Markov process, and therefore cannot be simulated with any of the methods described above.

Nevertheless, these events affect many cellular processes, including the dynamics of the gene regulatory network [18], and therefore need to be considered. The exact nature of the process may not be known, so constructing the explicit model may not be feasible or possible. Allowing reactions that produce their products an arbitrary time later in the simulation has two advantages. First and foremost, it allows the modeller to insert delays of arbitrary distributions even if the underlying process is not known. Secondly, it potentially removes many reactions from the system while not affecting its dynamics, speeding up simulations considerably [49].

To implement reactions that can have delayed products with potentially different delays, these products are placed on a "wait list" as a tuple (t_r, i, n), where t_r is the time at which the n molecules of S_i product should be released. The wait list itself can be implemented directly in the NRM with logarithmic time per addition or removal by adding products on the wait list as nodes of the indexed priority queue. Alternatively, it can be implemented to run alongside a DM implementation as a heap-based priority queue with the same runtime bounds using the following variant of the Execution step (paraphrased from [21]):

if $\tau < t_{min}$, where t_{min} is the earliest entry in the wait list then
    Perform the normal SSA Execution step
else
    Set $t = t_{min}$, and $X_i = X_i + n$.
    Remove the earliest entry from the wait list.
end if

2.4. Dynamic Compartments

In previous sections, simulation algorithms to simulate the time-evolution of a spatially homogeneous mixture of chemicals in a single reaction volume are discussed. However, many systems have dynamically-relevant inhomogeneities. It is thus of interest to have a stochastic formulation which accounts for some of these spatial effects. A first approximation, commonly used, is to divide the space into discrete regions called compartments, each of which assumed to be spatially homogeneous. In what concerns biochemical processes, the cell can be visualized as a hierarchy of compartments, each of which enclosed by a membrane. Each compartment may contain elementary molecules as well as other compartments. Examples of such structures in a eukaryotic cell include the mitochondria, the Golgi complex, and other organelles. In this scheme, the processes in a cell are viewed as sequences of discrete events such as a chemical reaction within a compartment, transport of molecules outside of, or into a compartment, and creation and dissolution of compartments.

Here, we make use of the P system formalism [22], which is an approach to simulate biochemical processes by making use of dynamically changing; nested compartments. As the presence of the nested compartments in P systems violates the assumption of homogeneous distribution of molecules of the SSA, we need to extend it [22].

For this, we first maintain the assumption that molecules are spatially homogeneous within a compartment, and therefore any of the variants of the SSA described above can be used to simulate the dynamics within each compartment. The transport of molecules from one compartment to another, as well as the creation and dissolution of compartments, are treated as reactions within the enclosing compartment. The multiple concurrent SSA simulations can then be integrated by using a NRM-like algorithm to determine in which compartment the next reaction takes place, effectively turning it into a ‘Next Compartment Method’.

3. METHODS AND APPROACH

We are interested in creating a realistic model of stochastic gene expression in *E. coli*. Therefore, we must first measure transcription dynamics in live bacteria. In this section, we describe the experimental procedures along with image processing and statistical methods used to obtain the required parameter values. This data from the experiments and theoretical knowledge from section 2.1 will be used in constructing a realistic model of gene expression, with particular emphasis in the kinetics of initiation.

We also measure the spatial kinetics of unwanted protein aggregates in live *E. coli* cells, which determines their partitioning distribution upon division. In this section, we also describe the experimental as well as the statistical methods to measure biases in partitioning of unwanted aggregates by these cells' poles. The data obtained is used in modelling the partitioning mechanisms of aggregates within the cells.

Finally, we present the design of a simulator that will be used to simulate the aforementioned models. The design is based on a combination of the efficient implementations of the SSA presented in sections 2.2 and 2.3, and incorporates the compartmentalization scheme from section 2.4. The compartment design is then extended to include several different physically-based molecule partitioning schemes.

3.1. Measuring the Kinetics of Transcription Initiation in Bacteria

To measure the kinetics of transcription, we follow a sequence of steps.

3.1.1. Experimental Setup

We use an *in vivo* single-mRNA tagging method in *E. coli* to detect, one molecule at a time, when these molecules are produced from a target gene in live cells. This method uses a fusion protein composed of MS2d, an RNA-binding capsid protein from the bacteriophage MS2, and the GFPmut3 fluorescent protein. In our measurements, the gene coding for this protein is under the control of P_{tetA} on a medium-copy number plasmid and activated before the target gene, ensuring that there is no shortage of reporter proteins for the target RNA. When co-expressed with a target RNA containing many binding sites for MS2d, this allows the individual target RNA molecules to be detected shortly after being produced. The expression of the target RNA is controlled by the *lar* promoter, also known as *lac/ara-1*, which was placed on a single-copy BAC plasmid [10]. Individual transcription events can be detected from a time-series of images under

the microscope, and the behaviour of the system was shown to be similar to that of the unlabelled system [13,41]. By detecting these events, it is possible to measure the time intervals between consecutive productions of RNA molecules under the control of *lar*, under various induction conditions [24,25].

The measurements were made at the Laboratory of Biosystem Dynamics, part of the Computational Systems Biology Research Group in the Department of Signal Processing, Tampere University of Technology. Cells for the experiments are grown in Miller LB medium and supplemented with appropriate antibiotics according to the specific plasmids. They are kept overnight at 37°C with aeration, and diluted into fresh medium and allowed to grow until the optical density reaches 0.5. Cells are then incubated with 100 ng/ml of anhydrotetracycline (*aTc*, from IBA GmbH) to attain full induction of the MS2d-GFP reporter. Approximately 60 min incubation allows sufficient production for RNA detection. Precise amount of inducers, such as 0-0.1% of L-arabinose (Sigma-Aldrich) and 0-1 mM of Isopropyl-β-D-thiogalactopyranoside (IPTG, Fermentas), are used to induce the target RNA. For complete activation of the *ara* system in the conditions where arabinose and IPTG are added, cells are pre-incubated with arabinose at the same time as aTc. These cells are grown and induced in the appropriate temperature.

3.1.1.1 Time-lapse single-molecule fluorescence microscopy

The cells are imaged under a confocal microscope in a thermal chamber set to 37°C. For imaging, a few μl of culture are placed between a cover-slip and a slab of 1% agarose containing LB along with the appropriate concentrations of inducers. When the reporter and target RNA are co-expressed, MS2d-GFP binds to the target RNA, forming a bright fluorescent spot. The RNA becomes visible during, or shortly after elongation [41]. Images of cells are captured from each slide every minute over two hours [13] using an inverted confocal laser-scanning microscope, Nikon Eclipse (TE-2000-U, Nikon, Japan).

3.1.2. Image Analysis

The time series of images are analysed in several stages, here described according to the order in which they are applied.

3.1.2.1 Drift correction

During the image acquisition process under the confocal microscope, for various reasons such as temperature changes, the objective or table tends to drift slightly in the *x* and/or *y* axes. As a result the same cells in adjacent frames are at slightly different positions, complicating the tracking of cells over time.

We eliminate this effect by using the cross correlation between consecutive frames to find the number of shifted pixels along the both axes. Since this process finds an integer amount of pixels, round- off errors can accumulate. This error can be corrected for

by comparing each frame with several preceding frames and taking the average shift. Once this process has been repeated for all the frames in the time series, the maximum area that remains common to all the frames is computed. All frames are then cropped to that area, considering the amount of drift that has occurred from the start. This pre-processing step allows future steps in the analysis pipeline to ignore the effect caused by the drift, making cell tracking over time easier.

3.1.2.2 Cell Detection

The first of the drift-corrected images is manually masked. During this process, the region that each cell occupies during the time series is painted a solid, unique colour. The following automated steps use the colour information to identify the cell region. The masking can be performed using graphical tools such as Paint.NET or GIMP. To account for cell division, masks are also created at each time-point a division occurs.

Once the masks are painted, a method based on Principal Component Analysis (PCA) is used to compute the dimensions and orientation of the cells using the pixel intensities within the masked regions of each cell. PCA is a statistical procedure to identify the axes of greatest variance in a multidimensional probability distribution. Using the intensity distribution as a probability distribution within each masked region, we obtain the principal components using PCA. The first principal component is taken to be the major axis while the second is taken to be the minor axis of the ellipse that represents the cell. The area of this ellipse is taken to be the cell area.

Other relevant data related of the cell to be used in a later stage of the analysis such as the mean cell intensity is also computed. This process is repeated for all cells in all images of the time-series. Cell lineages are reconstructed by assigning the cell that is closest to a divided cell in the previous frame as that cell's parent.

3.1.2.3 Spot Detection

After detecting the cells, we detect tagged RNA molecules (MS2d-GFP-RNA complexes) in each cell. We segment the MS2d-GFP-RNA spots with the kernel density estimation method (KDE) for spot detection proposed in [57]. The kernel density estimation, also known as Parzen window, estimates the probability density function over the image from local information. The method processes an image f by filtering it with a desired kernel as follows [57]:

$$\hat{f}(i,j) = \frac{1}{card(C(i,j))h} \sum_{(k,l)\in C(i,j)} K\left(\frac{f(i,j) - f(k,l)}{h}\right) \tag{11}$$

where h is the smoothing parameter or bandwidth, (k, l) represents pixel location inside the kernel, card is the cardinality of the set, and $K(u)$ is the kernel. We use a Gaussian kernel following the implementation in [58].

KDE classifies each pixel inside the cell as either from a spot (foreground) or from the cell region (background) based on its estimated density function. The classification is done by finding a local minimum point which is treated as the cut-off threshold be-

tween background and foreground. The MS2d-GFP-RNA spots are thus segmented from the kernel density estimated image, highlighting the spots.

Background-corrected spot intensities are then calculated by subtracting the expected intensity from the cell background from the total intensity in the spot region. These background-corrected intensities are summed for each cell to produce a time-series of total spot intensities. The image analysis process is illustrated in *Figure 3*.

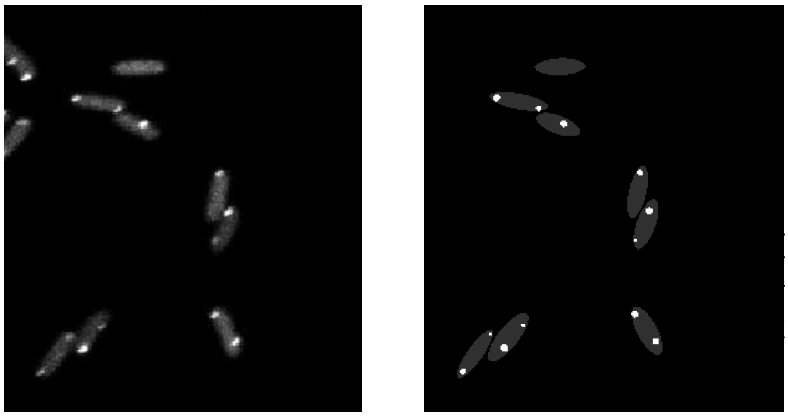

Figure 3: *A raw grayscale image (left) taken by confocal microscopy. MS2d-GFP RNA spots are visible. Same image (right) is shown after analysis. Cells are seen as grey ellipses and segmented spots are seen in white.*

3.1.3. Jump Detection

As described below, one of our aims is to determine intervals between transcription events (i.e. mRNA productions). We make use of the fact that, during the course of the measurements, target RNA molecules, once produced, are “immortal”. Namely, once tagged by the MS2d-GFP proteins, these target RNAs do not degrade and thus, we can assume that the number of target RNA molecules cannot decrease during a measurement.

Making use of this safe assumption, new RNA molecules are identified by fitting the time-series of background-corrected spot intensities to a monotone piecewise-constant function by least-squares [24]. In this procedure, the best-fitting curve for a given intensity time series is found by minimizing the sum of the squared difference between the measured curve and the fit curve, with the constraint that the number of spots cannot be greater than the number of mRNAs within the cell. The F-test is used to determine the number of terms in the piecewise function. The F-test requires that a higher order curve fits significantly better than others to justify its usage. That means we repeatedly reject lower order curves in favour of higher order curves with a p-value threshold 0.01. This process is illustrated in *Figure 4*.

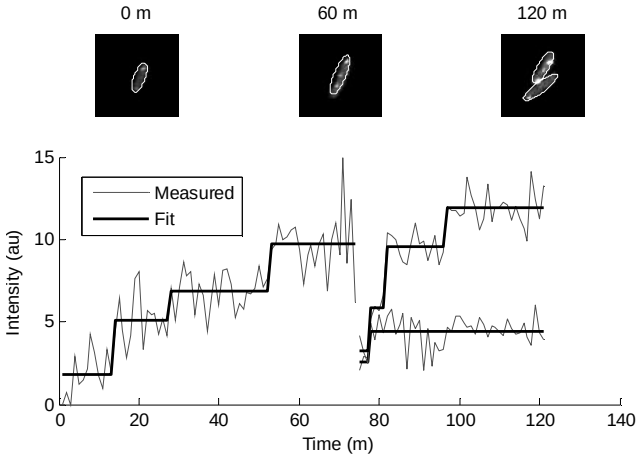

Figure 4: *Example time-series of a cell over two hours (top). Corresponding measured intensity time series and the intensity function fit by least squares (bottom).*

Jumps in the fit function are expected to correspond to the appearances of new RNA molecules. From these appearance times, we calculate time intervals between consecutive mRNA production events in each cell. Here, we restrict ourselves to only consider intervals entirely contained a cell's lifetime (i.e, we do not consider an interval if a cell division takes place between the two transcription events). *Figure 4* shows spot intensities in a cell up to 75 minutes (at this point, the cell divides). Following division, cells are still tracked, along with mRNA appearances within them. The plot for intensities of the two daughter cells is shown after 75 minutes. The figure shows two intervals in the parent cell, the first lasting 14 minutes and the second lasting 24 minutes. Using this procedure, the time interval distribution between productions of consecutive RNA molecules for each cell in the time-series can be computed.

3.1.4. Step Inference

We are interested in characterizing the durations of each of the sequential steps during transcription initiation. However, the time between visible production events is affected by another process: transcription elongation. We note that the mean elongation time does not affect the distributions, but a large variance in elongation can increase the variance of the distribution. Since transcription elongation takes a relatively short time compared to initiation [41], and has relatively low variance [59] when not affected by, e.g. sequence-dependent pause sites [5], we assume that the measured intervals between the productions of consecutive RNA molecules are not significantly affected by the elongation process. We thus consider the distribution of time intervals obtained as the duration of the transcription initiation process.

Furthermore, we investigate to find the number and durations of the sequential elementary steps in initiation [9-12,38]. To do this, we infer the properties of the steps by fitting the distribution of intervals between consecutive production events to a multi-step model, where each step is exponentially-distributed, as in [25]. For a given number

of steps *d*, the measured distribution is fit in the maximum likelihood sense to the probability density function of the sum of *d* exponential variables with different means. For statistically independent *d* steps with parameters $\mu = [\mu_1, \mu_2 ... \mu_d]$ inferred from N time intervals, the log-likelihood is:

$$L(\mu) = \sum_{k=1}^{N} \log \pi_d(\Delta t_k; \mu) \tag{12}$$

where Δt_k are the measured intervals for $k = 1 \ldots N$ and π_d is the probability density function for a sum of *d* exponential variables with means μ.

The probability density function for the sum can be calculated using the convolution of the probability density functions of the individual exponential random variables. The density functions for $d = 1 \ldots 3$ are [25]:

$$\pi_1(\Delta t_k; \mu_1) = \frac{e^{-x/\mu_1}}{\mu_1}$$

$$\pi_2(\Delta t_k; \mu_1, \mu_2) = \frac{e^{-x/\mu_1}}{\mu_1 - \mu_2} + \frac{e^{-x/\mu_2}}{\mu_2 - \mu_1}$$

$$\pi_2(\Delta t_k; \mu_1, \mu_2, \mu_3) = \frac{\mu_1 e^{-x/\mu_1}}{(\mu_1 - \mu_2)(\mu_1 - \mu_3)} + \frac{\mu_2 e^{-x/\mu_2}}{(\mu_2 - \mu_1)(\mu_2 - \mu_3)} + \frac{\mu_3 e^{-x/\mu_3}}{(\mu_3 - \mu_1)(\mu_3 - \mu_2)}$$

The number of steps is then selected using a likelihood ratio test between a *d* and a *d*+1 step model, rejecting a lower-order model in favour of a higher-order model with p-value threshold 0.01. The reliability of this method in distinguishing the duration of each step has been verified in [25]. The process of RNA production is thus considered to be composed of several steps with durations whose expected mean and standard deviations are given by $\mu = [\mu_1, \mu_2 ... \mu_d]$.

To assess the robustness of the inference procedure for an inferred set of mean step durations, we compute the standard deviation of the step durations inferred from time intervals that are sampled from the distribution of time intervals prescribed by the inferred model [60]. That is, for an inferred model with *d* steps, N simulated time intervals are sampled by summing d exponentially distributed variables with the same means as the inferred model, where N is the number of intervals obtained from the measurements. The inference process is then applied to this sampled distribution, and the standard deviations of each of the *i* longest durations are calculated, i.e., the variance of the shortest step is calculated from the distribution of the shortest inferred steps.

3.2. Assessing the kinetics of partitioning of unwanted protein aggregates

To assess the kinetics of partitioning of unwanted protein aggregates, we use the same method of tagging as in measuring the kinetics of initiation. Since every RNA molecule produced by the cells becomes bound by a large amount of MS2d-GFP proteins, the spatial dynamics of the complex likely differs from that of the unbound RNA [61]. The MS2d-GFP-RNA complexes can be tracked for several hours in live cells due to the substantially extended life time of the aggregate [41] allowing the observation of how these aggregates are partitioned in cell division [44].

We take two different measurements here. First, similar to the measurements for the initiation kinetics, we take a time series recording of individual cells. Images were taken approximately 7 min after induction by IPTG (67 minutes after aTc and arabinose), one every minute, over a period of approximately 2 hours. These time series recordings will be used to assess how the segregation of aggregates works, one molecule at a time. Second, we image populations of cells approximately 1 hour after induction by IPTG using a confocal microscope. We will use these images to examine the positions of the aggregates of many cells at once at a specific point in time.

3.2.1. Detection of cells and quantification of aggregates

In “population” images, we detect cells using a fully-automatic method from the raw images [62]. This method divides a grayscale image in three classes: background, cell border and cell region. It then exploits an iterative cell segmentation process that identifies and segments clumped cells based on size and edge information. The cell detection performance can degrade in regions where several cells are clumped together. This can be avoided by applying a threshold based on cell size and discarding the cells whose size goes beyond the threshold. Once the cells and their attributes have been identified, the MS2d-GFP-RNA spots are detected using KDE [57] as described in section 3.1.2.3.

Quantifying individual RNA molecules is not trivial since MS2d-GFP-RNA complexes can be co-localized [13]. The number of GFP molecules attached to an RNA molecule at any given moment can vary from 40 to 100 (70 on average) [41]. However, in general, the first peak of the distribution of intensities of many spots from cells on the same slide corresponds to individual MS2d-GFP-RNA complexes [13]. From that, the intensity of a single MS2d-GFP-RNA complex is obtained. Subsequent peaks in the distribution of intensities correspond to spots consisting of multiple MS2d-GFP-RNA complexes. The number of MS2d-GFP-RNA complexes in a spot is then estimated by normalizing the intensity of the spot by the intensity of a single MS2d-GFP-RNA complex [13].

3.2.2. Expected difference in number of aggregates between poles

We use |ΔN|, the absolute difference between the number of MS2d-GFP-RNA complexes detected in each pole of the cell, as a measure of the degree of bias of the partitioning distribution. We assume that the partitioning of the MS2d-GFP-RNA complexes between the two daughter cells follows the preferential partitioning scheme described in section 3.3.3. This partitioning distribution follows a biased binomial distribution with bias *p*. The expected |ΔN| for a given total number (N) of MS2d-GFP-RNA complexes in a cell can be calculated as follows:

$$E_{|\Delta N|}(N, p) = \sum_{k=0}^{N} |2k - N| \binom{N}{k} p^k (1-p)^{N-k} \tag{13}$$

We determine the bias of the partitioning distribution by finding the *p* such that $E_{|\Delta N|}$ matches the measured mean |ΔN| for all N.

3.3. Simulation Approach

Our simulation approach is based upon using a combination of the NRM, PDM and the delayed SSA to efficiently simulate the dynamic compartments described in [22], including delayed events. The overall simulation is driven by the NRM, since it is an extremely flexible discrete event simulation which can incorporate other simulation algorithms, termed sub-simulations. For example, the reaction system within each compartment is represented by its own NRM priority queue, which publishes a “next firing time” to the overall NRM. The indexed priority queue in the overall NRM provides the flexibility needed to add and remove entire sub-simulations (i.e. compartments) at runtime.

The main simulation loop consists of the following steps:

1. **Selection**: The event and the time of its occurrence is determined. This step runs in constant time since the next compartment in which an event occurs is stored at the front of the indexed priority queue, and the next event to occur in that compartment is stored at the front of its indexed priority queue.
2. **Execution**: The event is performed, moving time forward and modifying the state of the simulation according to the type of the event that occurred. Reaction propensities that depend on the changed state are flagged as *dirty*.
3. **Update**: The *dirty* propensities and putative reaction times are recalculated and the changes are propagated back up the data structures. This step takes O(logS U(S)), where S is the number of sub-simulations in the system and U(S) is the mean time required to update the dirty sub-simulations.

The runtime considerations of this simulation loop thus depend solely on the time complexity of the execution and the update step for each of the possible sub-simulations.

3.3.1. Compartment Representation

Compartments contain a subset of the reactants in a simulation, which interact differently with the rest of the system, often only reacting with other molecules in the same compartment. That is, reactant α in compartment P will react with other molecules in P, but not with reactants in compartment Q, including other α molecules. Since each compartment contains a set of molecules, each one will be assigned its own state vector.

To define what reactions are to be created when a given compartment is created, we introduce the notion of the compartment type. Every compartment is of a specific type, which contains a certain set of molecular species. We denote a molecular species α that is *contained* in a compartment of type P as α@P. Reactions in the model are defined as occurring within or between compartment types. A separate instance of each reaction occurring in type P will be created in each compartment of type P.

The compartmentalized systems we would like to simulate are naturally represented as a *hierarchy* of compartments. For example, we could run a detailed compartmentalized model of gene expression [40] within a dynamic population of cells. Compartments are therefore hierarchy organized, such that higher-level compartments *contain* lower-level compartments. Compartment types are similarly organized, creating a compartment type hierarchy. Inter-compartment reactions are allowed occur between compartments and their containing compartments. For the present simulation, direct inter-compartment reactions between compartments at the same level of the hierarchy are disallowed. The following sections detail the considerations made when simulating intra-compartment and inter-compartment reactions.

3.3.1.1 Intra-compartment Reactions

The reactions occurring within a compartment are intra-compartment reactions. Each compartment contains its own NRM priority queue, containing all the reactions that can occur within that compartment, and which publishes a “next firing time” to the overall NRM. The execution step of the reaction only needs to move the simulation time forward and modify the populations of the molecules that the reaction changes, and takes O(1) time. The update step, however, takes O(RlogM) time where M is number of reactions in the priority queue of that compartment and R is the number of those reactions whose propensities must be updated. We assume that in most cases, R does not grow with M (i.e. the reaction dependency graph is sparse), thus we do not take precautions against a dense dependency graph.

3.3.1.2 Inter-compartment Reactions

The reactions that span between the compartments are inter-compartment reactions. Since these are more complicated than intra-compartment reactions, we impose the restriction that these reactions may only span vertically across the compartment hierarchy (between parents and children), but not horizontally (between siblings). Therefore, a

child compartment can only affect a sibling indirectly by first changing the parent compartment.

To simulate a system with these reactions, we need to compute the propensities of all such reactions occurring in it. For this, we introduce the notion that a particular vertical reaction 'occurs' in the lowest level compartment in which it has a reactant or product. The propensities of vertical reactions are then calculated from the possible combinations of reactants in the compartment the reaction occurs in, and its containing compartments. This implies that there is a separate reaction for each lowest-level compartment that a vertical reaction reacts in. For example, if we consider a system with a compartment Q within another compartment P, the propensity a_μ that a molecule '*a*' in P will react with a molecule '*b*' in Q is given as.

$$a_\mu = X_{a@\mathrm{P}} X_{b@\mathrm{Q}} c_\mu$$

One straightforward way to implement vertical reactions would be to include putative reaction times for these reactions in the NRM simulation of the intra-compartment reactions that can occur in each of the compartments. The NRM running in each compartment will then select these reactions to occur with the correct timings. Since multiple reaction propensities in sub-compartments (child compartments in hierarchy) depend on a single number (the population of a molecule in the sub-compartments' common super-compartment), any change in the population of the molecule in the super-compartment will cause the NRM update logic to be called once for each sub-compartment. This will cause the update step to take O(C log M) time, an unacceptable linear scaling on the number of compartments C.

To avoid this linear scaling with the number of compartments, we factor out the population of the molecule in the super-compartment when computing the propensity of a reaction. The idea is similar to that of the PDM (described in section 2.3.5). Here, the vertical reaction occurring between the parent and the child compartments are considered as a separate sub-simulation of the parent compartment's NRM. This sub-simulation has its own time that passes as $X_{super}dt$, where X_{super} is the population of the molecule in the parent compartment. For a given reaction time in the simulation, the reaction with the earliest tentative firing time remains at the front of the priority queue of the sub-simulation and can be retrieved in a constant time. Any change in the population of the molecule in the parent compartment (X_{super}) then only requires the next firing time of the sub-simulation to be updated. The update of the firing time of any reaction within the sub-simulation can then be done in O(logC) time. The overall update time is then O(logC+logM), which is significant improvement over the previous approach.

3.3.2. Delayed Reactions

The delayed products of a reaction are inserted into a waitlist and are retrieved and reinserted in to the simulation when the actual simulation time crosses the delayed time. A separate wait list is created for each compartment, and is inserted as a sub-simulation of

the global NRM. The wait list itself is priority queue of delayed species. Insertions and removals from the queue take O(logW), where W is the total number of molecules on the wait list of a given compartment. Since W is expected to grow at most linearly with M, insertions and removals from the heap can be considered to take O(logM) time. The Execution step of the wait list sub-simulation then takes O(1) time since the next wait list event is readily available at the front of the priority queue, and the Update step takes O(logM) time. During another sub-simulation's execution step (e.g. a reaction in another compartment), elements may be added to the wait list of. This therefore imposes a minimum expected runtime bound on the execution step of Ω(logM) for those simulations as well, provided the number of elements added per step is constant.

The wait list for each compartment is also beneficial during the compartment destruction. When modelling a dynamic population of cells, the construction and destruction of the compartments can be a regular event. Since the wait list exists are distributed to the compartments, the compartments can be destructed without any global dependencies of delayed molecules within it.

3.3.3. Compartment Division and Molecule Partitioning

The division of cells is modelled by the construction of compartments during the simulation. When a division occurs, the original compartment (the parent) remains in the simulation as one of the daughter compartments, while a new compartment is created in the system for the other daughter. This is handled efficiently using the NRM to add and remove the compartment's reactions from the system.

The division of a cell also requires the partitioning of molecules within it to the newly created cells. The physical mechanisms behind the partitioning may differ between molecule species. Thus, upon division, we execute a "mock partitioning process" for each divided species describing the partitioning statistics. Several partitioning schemes and their mock processes have been described and their statistical properties examined in [7]. We implement all of these processes, as well as some additional ones. These processes are divided into three broad groups [7].

Independent Partitioning. This is a simple partitioning distribution where each partitioned molecule has an independent probability of ending up in either of the daughter cells. This type of independence can be realized by having a well-mixed cytoplasm, by having immobile molecules independently appear in either cell half, or by randomly picking the molecules and moving those to either cell half. This results in a partitioning scheme that is unbiased and the molecules after partitioning are distributed binomially.

Disordered Partitioning. In disordered partitioning, the variation in the partitioning system or the intracellular environment randomizes population levels between daughter cells more than independent partitioning. The following paragraphs describe several disordered partitioning schemes.

If cells divide such that the daughter cells do not have the same volume, we assume that the larger daughter is more likely to inherit more molecules than the smaller one. To implement this, we model the size asymmetry with a Beta distribution. The molecules are then segregated binomially depending upon the ratio of the daughter cell sizes. This partitioning results in significant differences in the daughter cells, and is referred to as *random size* partitioning.

If certain components or molecules in the cell form or exist in clusters, and the clusters of these components are divided independently during division, this will result in greater variance from division. For example, protein molecules or unwanted waste in bacteria are found aggregate and unevenly partition during division [14,63]. To model their partitioning during division, we first group the molecules into N clusters and then independently segregate the clusters into the daughter cells. This partitioning scheme is referred to as *clustered* partitioning.

Some molecules in the cell have been observed to move preferentially towards one of the cell poles. For example protein aggregates preferentially move towards the older pole [14]. This will result in a scheme where most of the molecules will end up in one of the daughter cells. This can be implemented by a biased binomial partitioning of the molecules such that the parent compartment likely retains more molecules than the newly formed compartment. After few generations this *preferential* partitioning scheme will result in older cells with very large number of molecules.

The partitioning scheme which would introduce the most variance possible during division is if all molecules are always partitioned into one of the daughter cells. We implement this by randomly selecting one of the daughter cell and putting all the molecules to it. We refer to this scheme as *all or nothing*.

Ordered Partitioning. In ordered partitioning, mechanisms in the cell interact directly or indirectly with each other to create a more even distribution of the molecules between daughter cells during division. The variance in partitioning is thus expected to be lower than independent partitioning. These mechanisms may be used by a cell as internal control mechanisms to make the partitioning more evenly distributed, or to compensate for disordered mechanisms.

With *volume exclusion* partitioning, cells can reduce partitioning errors passively if the segregating components occupy sufficiently large volume. A large amount of such molecules in one daughter will ‘push’ some of them into the other daughter prior to division, enforcing a more even partitioning. To implement this partitioning, we iteratively assign each molecule to the daughter cells based on the volume available in the daughter cells. Each time a molecule is assigned, its volume is subtracted from the available volume in the daughter cell. Care must be taken by the modeler to ensure that, e.g., the partitioned molecules do not occupy more space than the total volume of the daughter cell.

The variance can also be lowered if the molecules within a cell can bind to spindles, such as during mitosis in a eukaryotic cell, resulting in a s*pindle binding site* partition-

ing scheme. Each daughter cell will inherit the molecules that their spindles manage to pull into that cell during division. The implementation of this is very similar to that of volume exclusion. Here, we assign the molecules to the binding sites available in the daughter cells rather than available volume. The unbound molecules are then independently partitioned between the daughter cells.

In a *pair formation* partitioning scheme, molecules form pairs prior to division. When the cell divides, each of the daughter cells receives one of the molecules of the pair. This strategy reduces the partitioning error significantly. We implement this by probabilistically forming pairs of molecules when the division occurs. The pairs are then split with a certain probability, with one molecule of each successfully split pair going to one of the daughter compartment. The molecules that failed to make pairs and the pairs that fail to split are then partitioned independently to the daughter cells. Interestingly, if many pairs fail to split, this can lead to disordered partitioning as well, as a special case of clustered partitioning.

At the extreme end of ordered partitioning is *perfect* partitioning, each daughter cell gets exactly half of the molecules in the parent cell. It can be implemented by dividing the molecules into exactly half and then putting them to the daughter compartments.

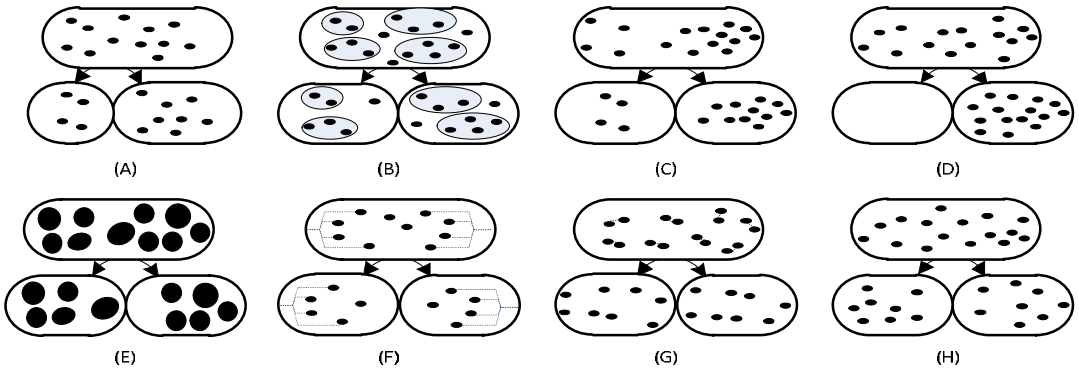

Figure 5: *Different partitioning schemes. (A) Random size (B) Clustered (C) Preferential (D) All or Nothing (E) Volume Exclusion (F) Spindle Binding site (G) Pair formation (H) Perfect*

4. SIMULATOR IMPLEMENTATION

We would like to construct a stochastic simulator, which we call *SGNS2*, which is easily extendable to incorporate new model features. At the heart of the simulation, therefore, we would like to have a very general structure which can accommodate new features. We are using the NRM as the overall simulation algorithm, which can be considered as a special case of a generic discrete event simulation. At the core of the simulator, therefore, we have implemented a general discrete event simulation, which can be extended to include non-SSA simulations if required in the future. The NRM and the associated structures described above are then implemented on top of the discrete event simulation. The simulator architecture is divided into five high-level modules, shown in *Figure 6*.

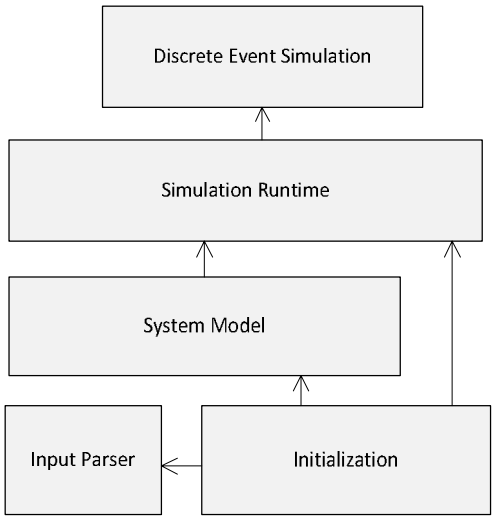

Figure 6: *Abstract architecture of SGNS2.*

The five modules perform the following functions:

1. Input Parser: This module consists of classes that read and recognize the SGNS file format which describes the model to be simulated.
2. Initialization: The input read from the parser is interpreted by the classes in this module to construct the internal representation of the model. These classes also initialize the simulation with the conditions and parameters that are input.
3. System Model: This module contains the classes used to internally represent the model to be simulated.
4. Simulation Runtime: The classes in this module are responsible for handling the runtime components of the simulation.
5. Discrete Event simulation: This module implements a generic discrete event simulation used by the simulation runtime.

Figure 7 shows a detailed class diagram of the simulator implementation. All classes in each of the five modules and their relationships are shown.

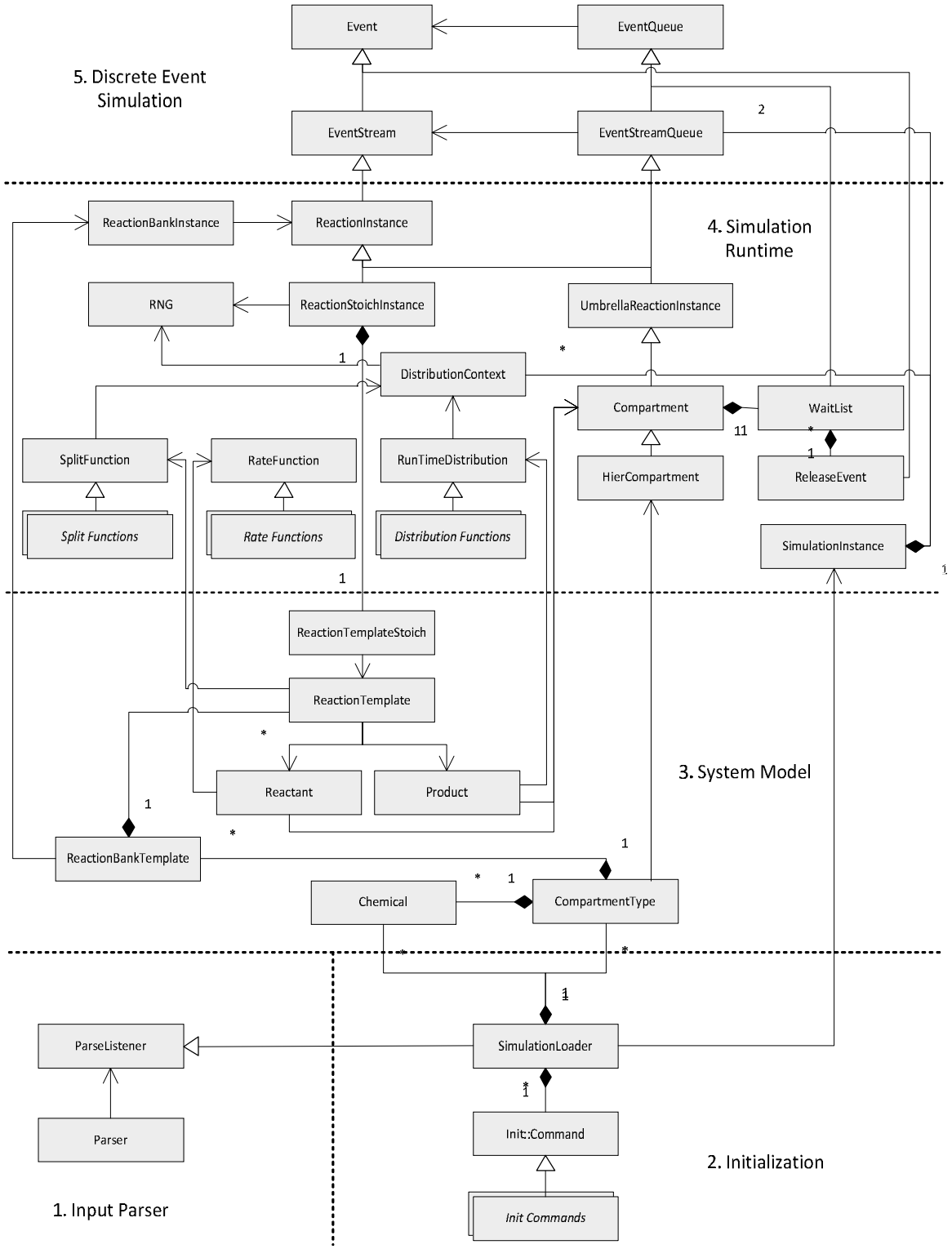

Figure 7: *Class diagram of the simulator implementation*

4.1. Input Parser

The parser module contains parsing functions to read the SGNS reaction file syntax and performs basic syntax checking. It uses the external "Lua" [64] library to provide a scripting environment to the user when constructing the reaction files. The `ParseListener` class acts as the parser-loader interface.

4.2. Initialization

The `SimulationLoader` class is responsible for interpreting the parsed input. From the input, it constructs the internal data structures representing the reaction model, stores the initial conditions for the simulations to be performed, as well as the overall simulation parameters (start and end time, sampling interval, output file and format, etc…).

`SimulationLoader` uses a flexible initiation command class (`init::Command`) so that new features of the simulation can be easily added. These commands perform tasks such as creating compartments, setting molecular populations, and filling the wait list. During simulation initialization, the class creates a new `SimulationInstance`, and executes all the initialization commands, which will set up the simulation to the starting configuration. This two-step initialization scheme is in place so that multiple simulations may be initialized and executed concurrently with a single call to the simulator.

4.3. System Model

The `ReactionTemplate` class in this module represents a generic reaction. It stores its stochastic constant and a list of reactants and products. The Reactant class represents a reactant in a reaction and stores the stoichiometry, the species in the containing compartment and rate function for the reactant. The rate function defines how the population of molecular species is incorporated into the propensity of the reaction. The Product class represents a product in a reaction and stores its stoichiometry, species in the containing compartment, and delay distribution. `ReactionTemplateStoich` objects are inserted into `ReactionInstances`, and provide the connection between the System Model module and the Simulation Runtime module. `ReactionBankTemplate` contains sets of `ReactionTemplates` which can be instantiated within or between compartments.

The class `CompartmentType` manages the species layout within a compartment type as well as the `ReactionBankTemplate's` to be instantiated within every compartment of that type and between parent and child compartments. It additionally stores the type name and its relation to other types. A special compartment type called Env is automatically created when the simulator is initialized, which represents the environment. All compartment types are contained in Env, either directly or indirectly, making it the root of the compartment type hierarchy. Additionally, this compartment type is the

default location of all reactants and reaction to occur. A single compartment of type Env always exists in the simulation, and new Env compartments cannot be instantiated or destroyed at runtime.

4.4. Simulation Runtime

The `SimulationInstance` class is the main class that is responsible for actual simulation. It contains the main NRM priority queue containing the high-level simulation events (compartment sub-simulations and wait list sub-simulations). Besides this queue, it contains an auxiliary queue to support actions such as sampling, saving, and others during the simulation. This separation of simulation events and supporting events ensures that identical simulations produce identical results, independent of the sampling and saving frequencies.

The `ReactionInstance` class is an abstract base class describing the basic operations that all reactions can perform. Besides tracking its next occurrence time (performed in the `Event` base class), this class contains the required functionality to listen to changes in reactant populations in `Compartment`'s.

The `ReactionStoichInstance` class is an implementation of `ReactionInstance`. It contains a Stoichiometry object which determines how the propensity of a reaction is calculated and what happens when the reaction is executed. The next reaction time is calculated here using the NRM τ generation and update formulae.

`UmbrellaReactionInstance` is the class responsible for the PDM-like factoring scheme described in section 3.3.1.2. It manages the next reaction firing time for a group of sub-reactions, which all have a common factor described by a Stoichiometry object. This class manages its own NRM priority queue (since it inherits from `EventQueue`) in which the sub-compartment's reactions are placed. This class is also insertable into the overall simulation NRM queue, or another `UmbrellaReactionInstance`, since it inherits from `ReactionInstance`.

The `Compartment` class represents a compartment. During the simulation, every compartment contains an array of all contained species populations. The layout of the species populations in a given compartment's state vector depends on the type of that compartment. This ensures that space is not allocated for species that do not exist in a given compartment. Reaction updates are performed when the population of a molecular species changes. For this, a dependency graph giving which `ReactionInstance`'s to update when a species changes is stored the in each compartment. Each compartment also contains a waitlist, which is implemented in the `WaitList` class with a binary heap-based priority queue, to store delayed molecules to be released in that compartment. `ReleaseEvent` is a class within the waitlist that holds the molecules that are to be released from the waitlist. Compartment is an implementation of `UmbrellaReactionInstance`, allowing it to contain the reactions which occur within it, simplifying the compartment construction and destruction operations.

`Compartment` alone does not contain compartment hierarchy or compartment type information so that the simulation can be extended in the future. Instead, this is the job of the specialized class `HierCompartment`. This class incorporates the idea of `CompartmentType`'s, and is responsible for managing the compartment's `ReactionBankInstance`'s.

The `RuntimeDistribution` class is an abstract base class of several classes which contain distribution functions which can be called to sample from several different delay distributions for the products in a reaction. This class stores two flexible parameters that can be used as parameters for different distribution functions such as Gaussian, exponential, and gamma. The distribution samplers are provided with access to the simulation's random number generator through `DistributionContext` struct. This struct provides the link between the low-level distributions and the higher-level `SimulationInstance`. Similarly, `RateFunction` is an abstract base class for classes which contain a calculation function that determines the effects that a reactant has on the propensity of its reaction, and is used to implement the propensity functions in section 2.2. There are two multi-purpose parameters to be used by the different rate functions.

The `SplitFunction` class behaves similar to the `RateFunction` and `RuntimeDistribution` classes. It is used to model the partitioning of molecular species that occurs during specific reactions (e.g. cell division). The various subclasses implement partitioning functions that correspond to the partitioning schemes. There are two parameters to define the distribution followed by the partitioning function and there are two additional flags. The first one is to identify if the partition is biased and the second one identifies if the partition is virtual. The virtual flag determines if the molecules get removed from the parent compartment or not.

4.5. Discrete Event Simulation

The `Event` class is a base class for all simulation events. It stores the time at which the event is to occur, and stores its index in its containing `EventQueue`. `EventQueue` implements the indexed priority queue of the NRM using a binary heap. `EventQueue` also stores the 'current time' of the simulation 'underneath' it. This is used to have different parts of the simulation run at different speeds and therefore exist at different times, which is necessary to implement the propensity factorization scheme described in section 3.3.1.2. `EventStream` and `EventStreamQueue` are specializations of Event and `EventQueue` for events which recur (e.g. reactions).

4.6. Using the Simulator

To use the simulator, the user creates an input reaction file in the SGNS format. The reaction file is often divided into four main parts: global simulation parameters, reaction parameters, initial populations, and reactions. Global simulation parameters define the

settings of the simulation and are represented by keywords such as *stop_time*, *readout_interval*, *show*/*hide* options, and *output_file*, among others. Reaction parameters are those that appear in the reactions' stochastic constants. These are enclosed within the keyword *parameter* using curly brackets. The next part of the input reaction file defines the initial populations of the species in the system to be simulated. The molecular species are initialized and are enclosed within keyword *population.* The reactions that drive the simulations are enclosed within *reaction* keyword. A complete example reaction file is shown in *Figure 8.*

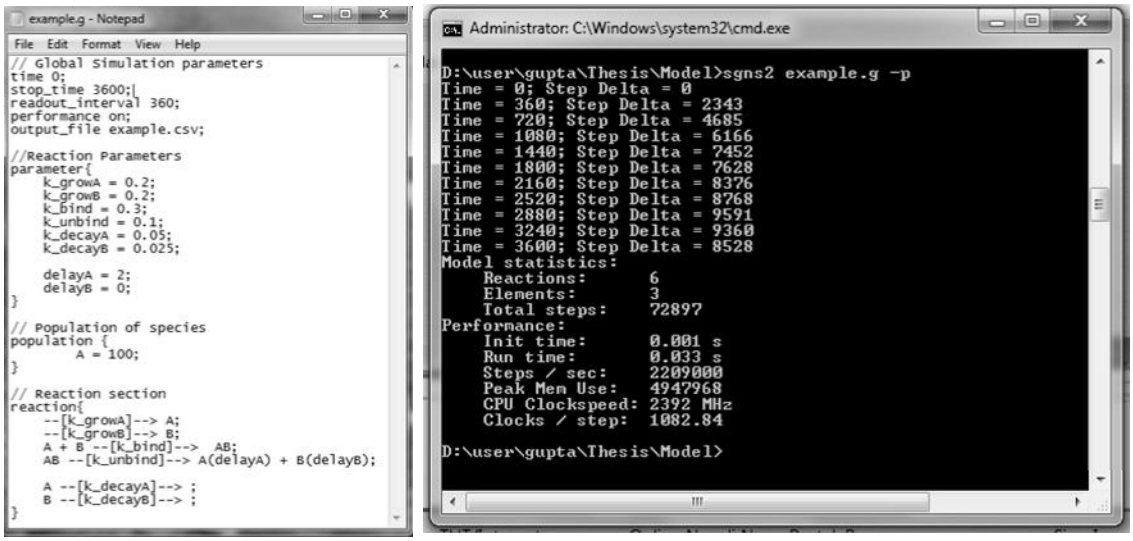

Figure 8: *An example reaction file (left). A command line run of the simulation (right). The time intervals (Time) along with number of SSA steps (Step Delta) occurred during each of the intervals are output during the simulation. The model and performance statistics are shown at the end of simulation.*

The input file can be written in any text editor such as Notepad or WordPad. We follow the convention that the reaction files have the extension '.g'. An example run for a reaction file is also shown in *Figure 8.*

5. RESULTS AND DISCUSSION

This thesis focuses to construct a realistic model of stochastic processes that lead to cell-to-cell diversity. To model these processes realistically, we need measurements of these process so that the models could mimic the actual process with accuracy. In the previous sections, we described experimental and statistical methods for the measurements of transcription initiation and bias in segregation of the protein aggregates. We now present results regarding the initiation kinetics of a promoter, from in vivo measurements of *E. coli*, one event at a time. Following that, we also present measurements of segregation kinetics of protein aggregates in live *E. coli*, one aggregate at a time. These measurements are then combined into a model that mimics the processes in a realistic way.

In previous sections, we also described our approach and its implementation for the simulator that would simulate such models. We use this simulator to simulate the models that we constructed from measurements. Then, we analyse the results of these models to characterize behaviour of cellular systems. Also, we test the efficiency of the simulator when compared to other contemporary simulator.

5.1. Kinetics of Transcription Initiation of the lar Promoter

To accurately model gene expression, we must have measurements of the distributions of time intervals between transcription events, so that the model can accurately reflect the stochasticity of this process. Further, this must be measured in several different induction conditions so that the repression and induction kinetics can be mimicked. We therefore measured the *in vivo* time intervals between transcript production events under five different induction conditions using the MS2d-GFP system. The results are shown in *Table 1*, including the mean transcript production rate and statistics on the time intervals between consecutive transcription events in individual cells.

Interestingly, the normalized variance of the time interval distribution is always less than 1. This is significant because if the process were Poissonian, i.e. events occurring randomly in time, this value would be 1 which is the normalized variance of the exponential distribution. This implies that transcript production from the *lar* promoter is less random than Poissonian, or sub-Poissonian. The change in normalized variance as induction is changed implies that the shape of the distribution is also changing. The distributions in three of these conditions are shown in *Figure 9*. As expected, these distributions look different from exponential (seen in the “1-step fit”).

Table 1: *Cells were induced by IPTG and L-arabinose as described in section 3.1.1. The mean number of RNA molecules produced by live cells per hour is shown for each condition (Mean RNA/h). The mean (μ), standard deviation (σ) and normalized variance (σ^2/μ^2) of the distribution of time intervals between subsequent production events in single cells are also shown. N denotes the number of such intervals observed in the experiments. In all cases, the distribution of these intervals was well-fit by a two exponentially-distributed step model with the means shown in the last two columns.*

IPTG (mM)	arabinose (%)	Mean RNA/h	*N*	Interval *μ* (s)	Interval *σ* (s)	σ^2/μ^2	Durations of steps (s)
0	0.1	0.3	108	1368	1128	0.68	(1122, 246)
0.1	0.1	1.0	71	1300	989	0.58	(976, 325)
1	0.1	1.6	343	965	698	0.52	(574, 391)
1	0.01	1.1	185	1483	819	0.30	(741, 741)
1	0	0.7	205	1587	1076	0.46	(793, 793)

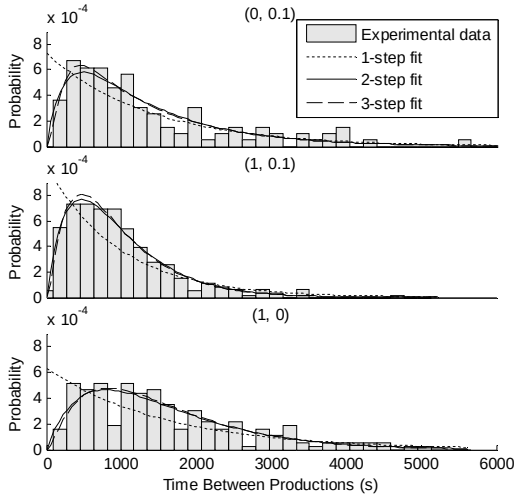

Figure 9: *Distributions of intervals between transcription events (gray bars) when induced by 0.1% arabinose (top), 1 mM IPTG (bottom), and both (middle). The best-fit models with one (dotted line), two (solid line) and three (dashed line) exponentially-distributed steps in initiation are shown.*

The observed intervals between the transcript production events were then used to infer the number and mean durations of the sequential steps in initiation (section 3.1.4). The result of the inference is shown in *Table 1* as well as in *Figure 10*. In all conditions, the likelihood ratio test (section 3.1.4) indicates that the interval distributions are better-fit by two exponentially-distributed sequential steps than one. A three-step model does not improve the fit significantly to reject the two-step model. This implies that we can accurately model the initiation kinetics with two exponentially-distributed steps. The error bars in the figure correspond to the variation in the inference method. This is important to determine if the changes in the durations of the steps are significant.

In *Figure 10*, as induction by IPTG is decreased, one step appears to become longer while the other does not change significantly. On the other hand, when decreasing induction by arabinose, at least one step increases in duration and the two steps become

similar in duration. Thus, we can conclude that IPTG and arabinose induce $P_{lac/ara-1}$ by different mechanisms. This interaction could then be modelled by modulating the rates of the appropriate steps according to the inference.

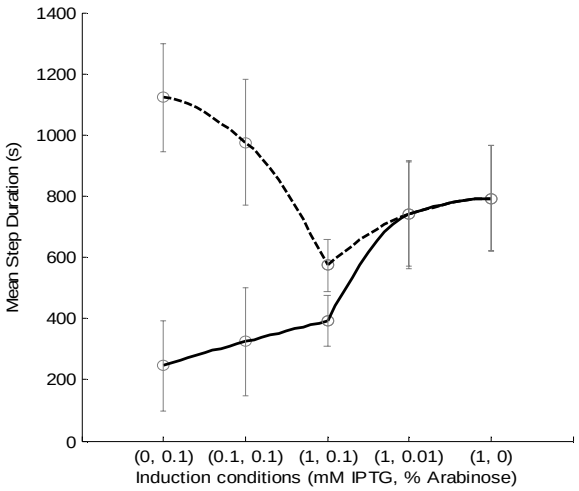

Figure 10: *Mean durations of the steps for each condition (circles). The standard deviations of the step duration inference for the same inferred means and the same number of samples are also shown.*

Here, the effects of IPTG and arabinose on the *in vivo* kinetics of $P_{lac/ara-1}$ were characterized. The model of two rate-limiting steps fit the data well in all conditions, though the durations of the steps vary widely between the conditions, and can therefore be used in a stochastic simulation of this process to drive initiation. Interestingly, our observations also indicate that the regulation of the sequential steps of $P_{lac/ara-1}$ in a graded and combined fashion permits the mean and fluctuations in RNA numbers to be tuned independently, since the variability of the interval distribution between transcription events will, in part, determine the fluctuations in RNA numbers.

5.2. Kinetics of Biased Partitioning of Protein Aggregates

To create an accurate model of aggregate partitioning during division, we observed the behaviour of individual aggregates in single cells. We used the same system as described above, since once the RNA has been bound by many MS2d-GFP's, it appears to be recognized by the cell as a protein aggregate [44]. Specifically, we imaged partially-induced (0.1 mM IPTG and 0.01% arabinose) cells under the microscope over a two hour period and observed the spatial dynamics of the produced aggregates.

First, we studied the location where aggregates first appear in the cell after being transcribed. By observing over 50 cells by eye, we found that the aggregates first appear near the cell centre, consistent with expected localization of the plasmid carrying the target RNA, as in [65]. Further, we measured the brightness of the aggregates appearing in the mid-cell region, and found it to be consistent with those aggregates consisting of a single tagged target RNA molecule each, as in [41].

Tracking the movement of newly produced aggregates, we found that all but 5% of them travelled to one of the poles during the observation period. Once an aggregate reaches a cell pole, it was observed that it tends to remain at the poles (except a few brief incursions to the mid-cell regions). Both of these observations are in agreement with observations in [41]. Further, even when only one aggregate was present in the cell, it travelled to and remained at a pole most of the time, which implies that this dynamics is not due to interactions between aggregates. *Figure 11* shows a typical sequence of images which exemplifies the observed spatial dynamics of aggregates.

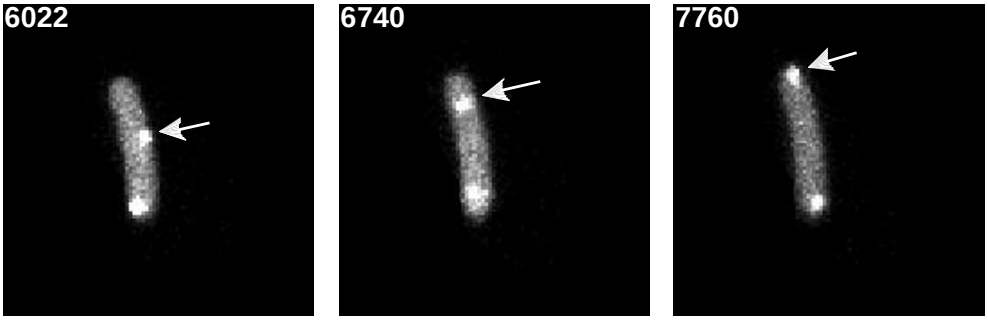

Figure 11: *Sequences of images of a cell with MS2d-GFP-tagged RNA molecules. The arrows point to an RNA spot (bright green spot) as it is formed in the mid-cell region, and moves from there to a pole.*

Next, we recorded the time at which individual aggregates were produced, as well as the pole to which they moved. The mean time between productions of consecutive aggregates is approximately 2000 seconds as these cells are only subject to half the maximum induction. The time to reach a pole was found to be much smaller than the mean interval between production events, indicating that newly created aggregates reach a pole prior to the creation of the next aggregate. Finally, a majority of the aggregates were observed travelling to the same pole in the cell. The fact that the time interval between transcription events is much larger than the time each aggregate takes to, independently, reach a pole indicates that the accumulation of the aggregates at the cell poles is not the result of aggregation with other aggregates. This is further supported by the fact about movement of lone tagged RNA in the cell towards the pole.

We then investigate the presence of the observed bias in the choice of pole. In *Figure 12*, the progression in time of the mean difference between the numbers of aggregates in each pole of each cell (denoted by $|\Delta N|$), summed over all cells for each time point is shown. For comparison, it is also shown how this quantity would vary over time, assuming binomial unbiased partitioning of aggregates by the two poles ($p = 0.5$), a totally biased partitioning ($p = 1.0$), and finally a partitioning that follows a binomial distribution with $p = 0.85$ towards one of the poles, which was found to fit the measurements well. In all analytical estimations, we assume the same total number of aggregates as those detected in the measurements at each moment in time.

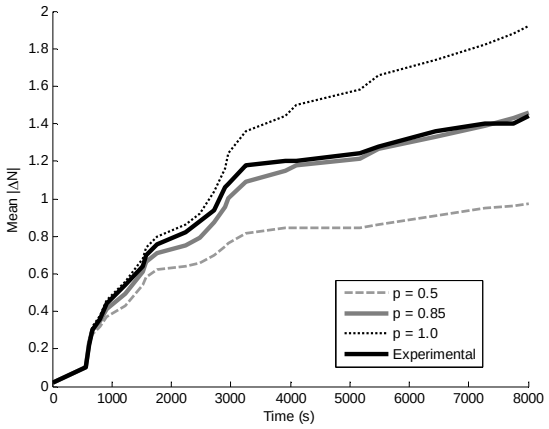

Figure 12: *Difference over time between numbers of aggregates in each pole (|ΔN|) averaged over 50 cells, each of which produced at least 1 spot during the observation period.*

We next investigate the effects of this process at the population level. From images of a population of 367 cells, 60 minutes after induction, we detected aggregate locations within cells using the methods from section 3.2.1. *Figure 13* shows the distribution of the distances of all detected aggregates from the centre of the cells, normalized by half of the cell length. The distribution is bimodal with a strong peak centred at 0.8, showing that most aggregates are located at the poles. A small "*peak*" exists at the mid-cell region, likely due to the partially-formed target RNA molecules visible while being transcribed and anchored to the plasmid [41]. This spatial distribution is striking similar to the one reported in [14], where the authors described the biased segregation of protein aggregates as a waste disposal mechanism. Further, the distribution is consistent with the spatial dynamics described above.

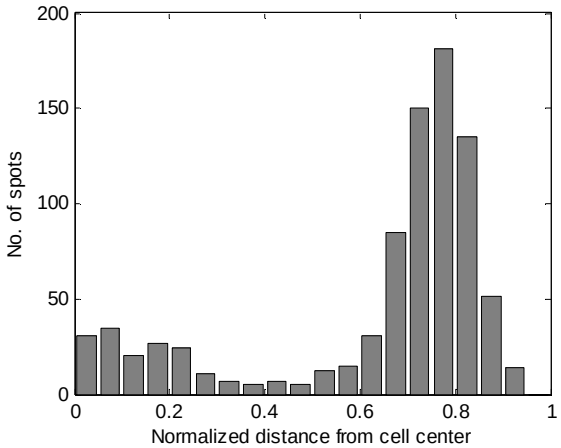

Figure 13: *Distribution of the distances of MS2d-GFP-RNA spots to the center of the cells, normalized by half of the cell length*

The mean difference in aggregate numbers between poles in the population, 60 minutes after induction as a function of the number of aggregates in each cell (N) is

shown in *Figure 14*. The expected value of |ΔN| as a function of N, assuming binomial partitioning of aggregates by the two poles, with a bias of 0.5, 0.85 and 1.0 is also shown for comparison. The results are in agreement with the temporal measurements since, again, a good fit is obtained with a highly biased binomial partitioning (p = 0.85). The results also show that the bias is independent of the number of aggregates in a cell, as would be expected if the bias results from asymmetric segregation of aggregates.

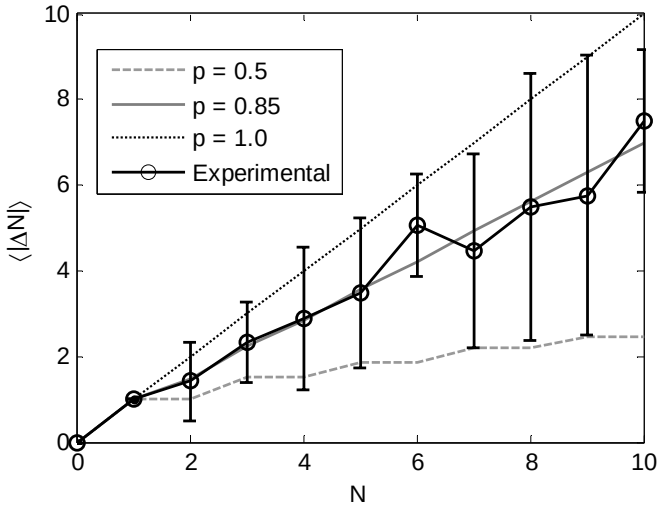

Figure 14: *Mean difference between the total number of aggregates in each pole of individual cells (⟨|ΔN|⟩) (solid black line) as a function of the total number of aggregates in the cell (N). Errors bars show the standard deviation. The expected ⟨|ΔN|⟩ assuming an unbiased binomial partitioning (p = 0.5, dashed line), a biased binomial partitioning (p = 0.85, solid gray line), and a totally biased partitioning (p = 1.0, dotted line) are shown for comparison.*

The MS2d-GFP tagging method was used to observe the kinetics of the protein aggregate disposal mechanism at the single event level as it allows observing individual tagged RNA molecules. Since MS2d-GFP-RNA complexes are not native to *E. coli*, it may be that the cell recognizes them as an undesirable substance and results in the behaviour as observed in [14]. Also, since the aggregates are segregated independently and probabilistically, this process can therefore be modelled by binomially partitioning of aggregates in the cells upon division, using a bias of p = 0.85.

5.3. Model of Cell-to-Cell Diversity in RNA Numbers

Using the results from section 5.1 of the promoter initiation dynamics of P_{lar}, and the partitioning schemes from section 5.2, we next build a realistic model of gene expression in a dynamic cell population. The model is based upon the modelling strategy put forward in [66]. The reactions involved in the model are:

$$Pro_C \xrightarrow{k_{cc}} Pro{\cdot}RNAP_C \quad (14)$$

$$Pro{\cdot}RNAP_C \xrightarrow{k_{oc}} Pro_C + RBS_C \quad (15)$$

$$RBS_C \xrightarrow{k_{trans}} RBS_C + P_C(normal:\ \tau_p, \sigma_p) \quad (16)$$

$$RBS_C \xrightarrow{k_{d\_rbs}} \emptyset \quad (17)$$

$$P_C \xrightarrow{k_{d\_p}} \emptyset \quad (18)$$

$$\emptyset \xrightarrow{k_{growth}} l_C \quad (19)$$

$$indpart{:}l_C + part{:}RBS_C + part{:}P_C \xrightarrow{k_{divide}} NewCell + {:}l_C + {:}\ RBS_C + {:}P_C + Pro_C \quad (20)$$

Reactions (14) and (15) model transcription initiation, and correspond to the two exponential steps observed in *Table 1*. Together, these reactions fully account for the distribution observed in *Figure 9*. Initially, we build this model assuming the full induction case in *Table 1*, though the model is easily adapted to other induction conditions. Reaction (14) models the closed complex formation, where an RNA polymerase binds to the promoter, represented as Pro_C, and forms $Pro{\cdot}RNAP_C$. We can freely choose one of the two inferred rates in *Table 1* as the rate of this reaction since it does not affect the result of our model. This $Pro{\cdot}RNAP_C$ then completes the transcription initiation process, releasing the promoter along with a Ribosome Binding Site (*RBS*) in the compartment, as shown in Reaction (15). For this reaction, the inferred rate that was not selected for the closed complex is chosen. The completion of transcription initiation is followed by translation, here modelled by Reaction (16) whereby proteins (P_C) are synthesized. The release of proteins is delayed to model protein elongation and folding, and is modelled by a normal distribution with mean τ_p and a standard deviation of σ_p. The lifetimes of the RNA molecules and the proteins are controlled by reactions (17) and (18), respectively. As the mRNA and protein have a certain life span, these reactions are important while modeling these systems.

Reaction (19) models the growth of the cell, by continually incrementing l_C, which represents the cell's length. When the cell has reached twice its original size ($l_C \geq 2000$), the division reaction (20) is allowed to occur. The time between divisions is tuned by modifying k_{growth} such that it matches the division time observed in the experiments of *Figure 4.* Every time a division reaction occurs, a new compartment is created and the contents of the parent compartment are transferred to the new compartment according to the partitioning scheme selected. In reaction (20), 'part' refers to an arbitrary partitioning scheme for the proteins P_C and RBS's RBS_C whereas it is independently partitions the length l_C (represented as 'indpart'). In the products side of the reaction, *NewCell* creates a new compartment due to division. The symbol '*:*' represents the partitioned molecules that need to be placed in the new compartment. For example, $:l_C$ is the partitioned cell length that is assigned to the newly created compartment. This division reaction therefore results in a new compartment will receive either all or none of the proteins and

RBSs in contrast to the length which it is likely to get halved. It also needs to be mentioned that these splits do not affect the reaction's propensity.

Realistic parameter values for the reactions above are given in *Table 2*, along with references.

Table 2: *Parameters for the gene expression and cell growth model. All rates are in* s^{-1}.

Parameter	Value	Source
k_{growth}	1/45	From observed cell division time, as shown in *Figure 4*.
k_{cc}	1/574	One of the inferred rate limiting steps of fully induced (1, 0.1) case, from *Table 1*.
k_{oc}	1/391	One of the inferred rate limiting steps of fully induced (1, 0.1) case, from *Table 1*.
k_{trans}	1/3	[40]
k_{divide}	1	Once k_{growth} reaches division point, division follows.
τ_{p}	7 minutes	[67]
σ_{p}	5	[40]
$k_{\text{d\_rbs}}$	0.011	[40]
$k_{\text{d\_p}}$	0.00003968	[40]

To demonstrate how the model may be used to study cell-to-cell diversity with different partitioning schemes, we now simulate the model under several conditions with the simulator constructed previously. *Figure 15* shows the time-series of protein levels from two such simulations that start with a single cell at the beginning. The protein level shows a somewhat high variance for the independent partitioning scheme. In the case of the random size partitioning scheme, however, where the cells at the end can contain almost any number of proteins. This produced much larger variance in protein numbers. In bacteria, as protein numbers follow the RNA numbers [19,28] and the phenotype depends on RNA[27], the cells at the end of the series for random size partitioning are thus expected to be more phenotypically diverse than for independent partitioning.

To investigate the effects of different partitioning schemes, we simulated the protein levels in cells that divide the proteins with different schemes, and measured the distribution of protein levels across the population after 1000 minutes of simulation. The results are shown in *Figure 16*. In this case, the stochasticity of transcription and translation causes the distributions to be similar for the ordered and independent partitioning schemes. With disordered partitioning, the distribution changes shape, becoming more

and more variable. The extreme case of the 'all or nothing' partitioning results in an almost geometric distribution.

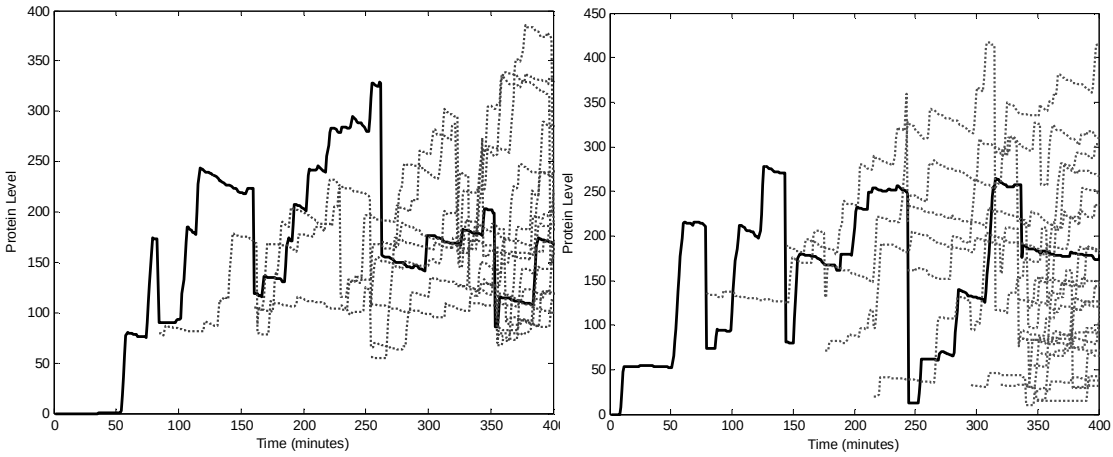

Figure 15: *Time-series showing protein level of a single cell (black solid line) and its daughter cells (dotted lines) with independent partitioning in division (left) which results in less variance in protein level among all cells and random-sized partitioning in division (right) with the size ratio following a Beta distribution with parameters $\alpha = 2$ and $\beta = 2$, which results in increased variance in protein levels among all cells.*

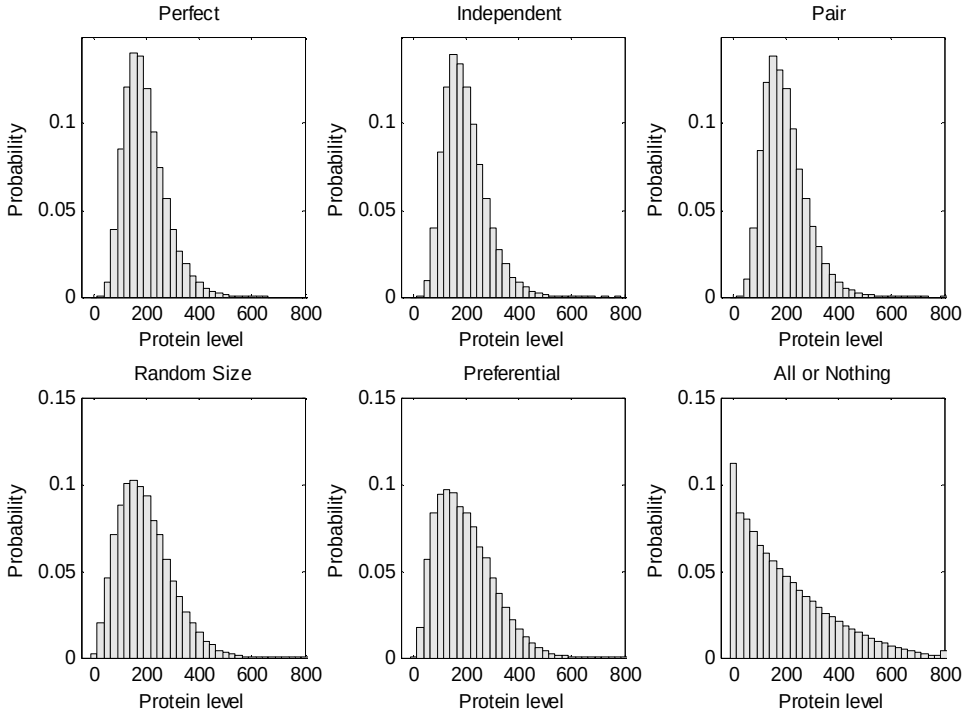

Figure 16: *Distribution of protein levels with different partitioning schemes at t = 1000 minutes. Partitioning schemes with variance lower than or equal to unbiased binomial distribution (includes ordered and independent partitioning)(top). Partitioning schemes with variance higher than binomial distribution (disordered partitioning)(bottom).*

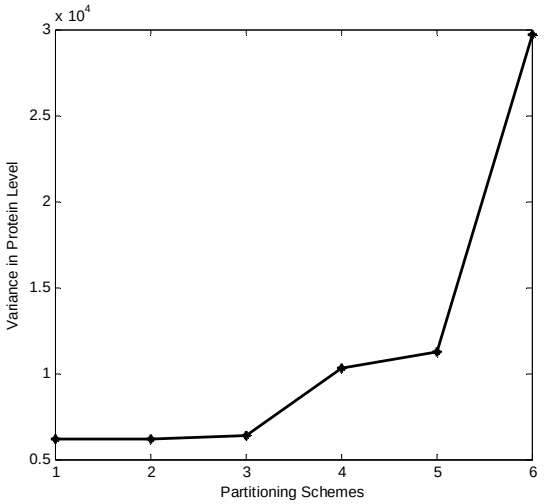

Figure 17: *Variance for partitioning schemes of Figure 16. Schemes 1, 2, 3 are perfect, independent and pair partitioning, respectively and have low variance. Schemes 4, 5, 6 are random size, preferential and all or nothing partitioning, respectively and have an increased variance.*

Since all distributions of protein numbers in *Figure 16* have the same mean, we use the variance as a measure of the variability of protein levels, shown in *Figure 17*. The distribution of protein levels for independent and ordered partitioning (Perfect and Pair partitioning) are found to have very similar, low variance. It can thus be concluded that the division involving such schemes result to individuals with a low variance in protein numbers and therefore results in generation with a more or less similar phenotype.

The distributions of protein levels for disordered segregation are remarkably different. The variance increases for the random sized and preferential partitioning and has the highest value for the all or nothing partitioning scheme. Disordered partitioning can clearly introduce a large amount of variance beyond the variance of gene expression.

By coupling gene expression models as the one described here with regulatory interactions, as in [66], gene regulatory networks can be constructed and simulated with the same tools as used here. In future studies, it would be interesting to observe what effects, if any, the different partitioning schemes have on the dynamics of genetic circuits. For example, disordered partitioning is expected to ‘destabilize’ the noisy attractors of genetic switches [68], and may affect the robustness of the periodicity of genetic clocks as well.

5.4. Simulating Ageing Processes in Cell Populations

We now construct a model to simulate the ageing process in *E. coli*, as a result of the biased aggregation of unwanted substances by the cell poles [15]. We use the measurements above (section 5.2) to set the bias by which aggregates tend to move towards the older pole in our model. For simplicity, we only mimic the aggregation and division, and ignore other cellular processes. The reactions composing this model are:

$$\xrightarrow{k_{aggregate}} \quad Aggregate_C \tag{21}$$

$$*Aggregate_C\ (invhill) \xrightarrow{k_{growth}} l_C \tag{22}$$

$$indpart{:}l_C + preferential(pbias){:}Aggregate_C \xrightarrow{k_{divide}} NewCell + {:}l_C + Aggregate_C \tag{23}$$

The aggregate generation is represented by reaction (21). Reaction (22) represents the growth of the cell, which is affected by the amount of aggregates in the cell (due to 'invhill' function). As the aggregates slow down the increase in length they affect also the division time. Cell division is modelled in reaction (23). Following its birth, once the length of the cell doubles, it divides, preferentially partitioning the aggregates, causing the 'newer' daughter to likely receive fewer aggregates. The parameters in this model are shown in *Table 3.*

Table 3: *Parameters for the cell ageing model.*

Parameter	Value	Source
k_{growth}	1/45	From observed cell division time, as shown in *Figure 4*.
$k_{\text{aggregate}}$	0.0019	Chosen arbitrarily to fit plot in [15]
k_{divide}	1	Once k_{growth} reaches division point, division occurs quickly.
pbias	0.15	Older cells give away (1-0.85) of the total molecules as measured in section 5.2

We simulate the model to study how aggregates affect a cell's growth rate and thereby its division time. As described in [15], the cells with older poles show a decreased growth rate, which results in a decreased offspring production, and an increased incidence of death. We use the model to replicate findings in [15]. We plot the division time of each cell for nine generations and show its lineage in *Figure 18*. We note that this plot is strikingly similar to the one in [15], though mirrored, since we plot division time rather than growth rate.

The effect of aggregates in the cells with older poles is seen in *Figure 18*. The far right cell, which is the oldest parent, has a higher division time and does lead to a ninth generation offspring in our simulation time. It is also evident that the cells getting new poles divide in similar times. We can therefore conclude that this model mimics the ageing process reported in [15].

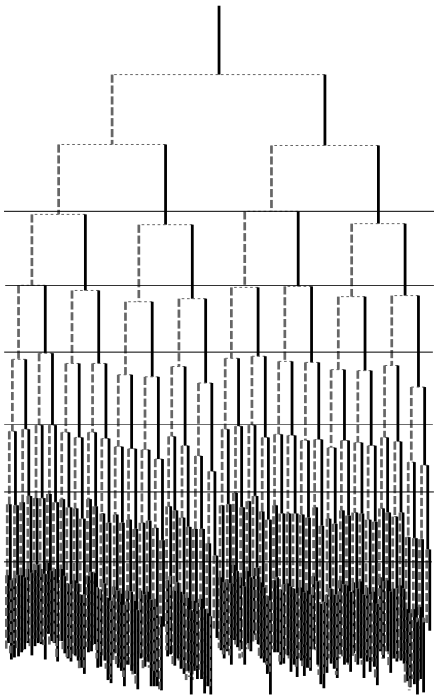

Figure 18: *Plot showing the division times of a lineage of cells starting from a single parent. Cells with older poles (vertical solid black line) are always on the right side of the cells with new poles (vertical dotted grey line) after division. The first division time (horizontal solid black line) for each generation is also shown.*

Further, by adding reactions associated with other cellular processes to this model, we can further characterize the behaviour of the cell. Also, it would be interesting to see the combination of this model with the gene expression model.

5.5. Simulator Performance

We have built a simulator that is capable of simulating the previously described stochastic models of gene expression and of ageing. Here, we analyse the efficiency of our implementation of the simulator. We investigate the performance of our simulator when compared to *Infobiotics Workbench* [69], which is, to our knowledge, the only other simulator which supports the same kind of compartment scheme as we do. The workbench is a multi-compartmental stochastic simulator based on the SSA for multi-cellular systems. Although it has a user friendly front-end that allows modelling a system, analysing and visualising the results, we use only its core simulator (mcss) to avoid any complexity brought up by the communication of friendly user interface and actual simulation. In addition, we compare our NRM-based simulator against an implementation of *SGNS2* using the LDM.

First, we compare performance using a simple gene expression model for a single cell without growth and division, but with auto-regulation. This simple model has six reactions for each gene which also includes reactions for degradation and binding of the protein product to the promoter. The model is constructed separately in *SGNS* format

and for *Infobiotics workbench*. To verify that the models built for these simulators are equivalent, in *Figure 19* we show the output of the simulation performed by both simulators. The output behaviour of the model with the two simulators is indistinguishable.

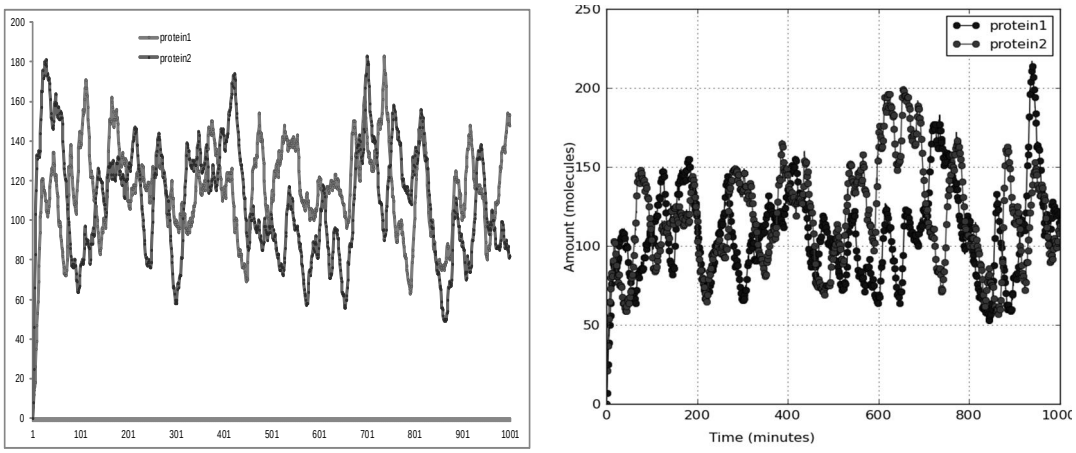

Figure 19: *Time-series of protein levels using models with 2 genes. Model was simulated using SGNS2 (left). The same model was simulated using Infobiotics Workbench (right).*

To test the performance of the simulators, we simulated models with differing numbers of self-repressing genes, and measured the amount of time required to complete the simulation. Increases in the number of genes increase the number of reactions in the simulation. Results are shown in *Figure 20*. For the first few increases in gene count, the performance for each of these simulators seems similar. However, when the number of genes exceeds five, there is a remarkable difference in the performance curves of *SGNS2* and *Infobiotics*. The time complexity of *Infobiotics* increases in an apparently quadratic manner with the number of reactions, and is clearly not as efficient as *SGNS2* for large systems, e.g. more than 100 reactions. To model even moderate systems in detail, we must consider more reactions than this. We were unable to run *Infobiotics Workbench* when number of reactions exceeded 1800, while this test successfully completed for *SGNS2* in 4.16 seconds.

Interestingly, there is no significant difference in performance between the two *SGNS* implementations for number of reactions up to 300. We therefore simulated the model with up to 1500 reactions with the LDM and NRM implementations of *SGNS2*. *Figure 20* also shows the comparison between these two implementations. It is clear from this figure that for a system with many reactions, the implementation with NRM is slightly more efficient than the one with LDM, thought they both scale equally well. This justifies our selection of NRM for the implementation of the simulator, which is, in addition, a very flexible discrete event simulation.

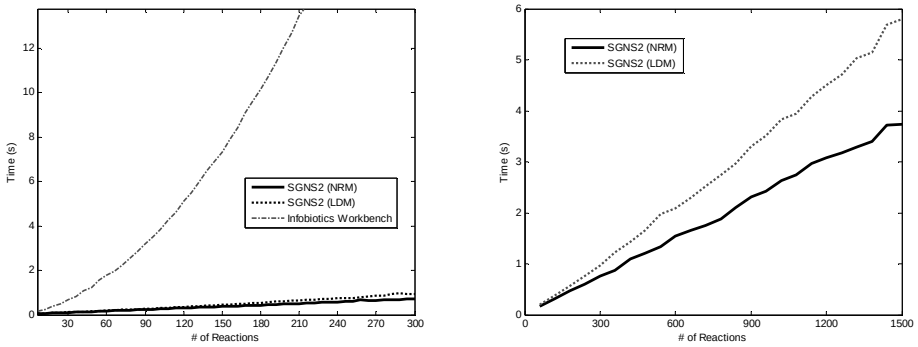

Figure 20: *Comparison of performance of NRM based SGNS2, LDM based SGNS2 and Infobiotics Workbench (left). The time complexity of the Infobiotics workbench looks like O(N^2), where N is number of reactions. Both the SGNS performs significantly better than Infobiotics. Comparison of performance of NRM based SGNS2 and LDM based SGNS2 with an increased number of reactions (right).*

We are further interested in comparing with *Infobiotics Workbench* from the compartments perspective. For this, we create compartments which hold the reactions of the self-repressing gene model. We tested the performance of the simulators by increasing the number of compartments instead of the number of reactions. The results are shown in *Figure 21*. We observe a slightly different scaling than our previous test. The time complexity of all the simulators appears to grow as O(NlogN), N being the number of compartments. However, differences in performance still remain. As in case of the number of reactions, NRM based *SGNS2* performs the best among the three. Although the scaling based on number of compartments is improved *for Infobiotics Workbench*, the constant remains large compared to *SGNS2*.

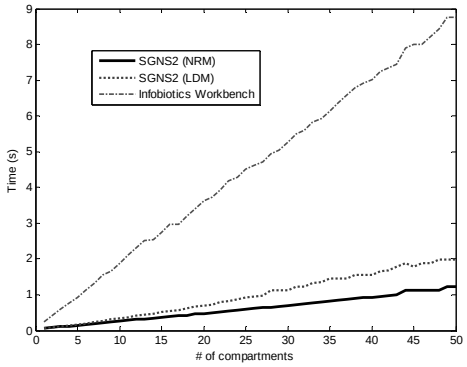

Figure 21: *Comparison of performance of NRM based SGNS2, LDM based SGNS2 and Infobiotics Workbench based on increasing number of compartments (N). The time complexity of all the simulators look like O(NlogN) although the NRM based SGNS2 perform significantly better.*

Therefore, the implementation of NRM based *SGNS2* can be considered as an efficient implementation of the SSA with the feature of dynamic compartments and various partitioning mechanisms. *Infobiotics Workbench* had been found to perform better than most of the current simulators [70]. As the performance of *SGNS2* is better than it, we can consider *SGNS2* to be better than others as well.

6. CONCLUSION

In this thesis, we have proposed a novel simulator for realistic models of gene expression and of stochastic partitioning of molecules in a dynamic population of dividing cells. For this, we have made measurements to obtain realistic parameter values for the models.

Specifically, we first measured the distribution of time intervals between production events of a promoter in live *E. coli* cells. From these distributions, we then inferred the number and mean durations of the sequential steps in initiation. We used these to derive realistic parameter values for a model of gene expression. The measurements further inform on the variability of the intervals between transcript productions in all induction ranges. With this data, the models can be extended to incorporate regulation accurately. In the future, similar investigations for different promoters are required to determine realistic ranges of parameters values. Also, some of the parameters will need to be validated using an independent method as it might be that the MS2 system perturbs the natural system in an unknown fashion. Unfortunately there are yet no such independent methods available regarding time intervals between transcription events.

A different measurement was performed to characterize the kinetics of segregation of protein aggregates in *E. coli*. We observed that the aggregates segregate to the cell poles by a mechanism that acts independently on individual aggregates. Further, we observed that the choice of pole is probabilistic and biased. The partitioning distribution was found to be well-fit by a binomial distribution with a bias of $p = 0.85$. Altogether, this implies that the process can be modelled with a preferential partitioning scheme. Since the accumulation of protein aggregates has been linked to ageing in these organisms, we used this to construct a model of cell ageing. While many features were extracted from these measurements, some questions remain unanswered about the segregation mechanism which would be needed to construct a more complete model. For example, it would be of great interest to determine whether the bias is similar or not for all unwanted molecules. This could be tested by observing the segregation of different fluorescently tagged proteins, such as *tsr*-Venus [35].

When developing the models of gene expression and aggregate segregation using the parameters that were measured experimentally, we found that this could not be achieved with existing simulators. For example, we found that it is necessary for the simulator to differentiate between physically identical molecules such as two identical RNA molecules. In short, the algorithms need to account for spatial location. To cope with this without losing the ability to model stochasticity accurately, we introduced the

concept of dynamic compartments. Similarly, to model partitioning of cellular components (e.g. molecules) in cell division and their effects in cell to cell diversity, we introduced the concept of partitioning schemes during cell division. We implemented and simulated various partitioning schemes and studied their effect on cell-to-cell diversity in RNA and protein numbers.

When developing the simulator and its several features, we emphasized efficiency. With that aim, the simulator uses the NRM along with the delayed SSA and PDM. The simulator was shown to perform more efficiently than a contemporary simulator with similar aims, *Infobiotics Workbench*. Additionally, as shown, the simulator is scalable and multi-purposed. Regarding the latter, we note that it can be used, as is, to model more complex systems than those used as test-cases. Regarding the former feature, we showed that the simulator is capable of simulating models that range from single expression in a bacterium to model a dynamic population of cells with subcellular compartments. To allow scalability and multi-usability the simulator was implemented using template classes which support modelling different kinds of events.

At present, the main limitation of this simulator is that it does not support horizontal reactions between the compartments. An initial implementation of these kinds of reactions could be done by assuming that they can occur between all pairs of sibling compartments. This would result in a C^2 amount of reaction instances to keep updated. Another limitation of the present simulator is that elements on the wait list are not partitioned during division. Thus, if a protein takes a very long time to fold, for example, this will effectively introduce a bias in protein production towards the parent compartment. To solve this, the wait list would have to be scoured for the partitioning molecule during division, an O(W) operation, where W is the number of delayed events in the wait list. We leave these to a future implementation.

Also, the efficiency of the simulator can be improved further. If the simulation is divided into a Markovian part (i.e. pure SSA), and a non-Markovian part (i.e. delayed reactions), we can then simulate the Markovian part with the SSA-CR, while the delayed parts follow the current implementation. This would reduce many of the logarithmic steps in the simulation to constant-time operations. Another improvement from the usability and extendibility point of view would be to implement a generic partitioning function using a Lua call-back, rather than hard-coded functions, allowing the user to introduce any partitioning function that can be expressed in Lua.

The simulator developed in this thesis can now be used to investigate, in general, various processes, ranging from viral infections of cell populations to cell-to-cell phenotypic diversity resulting from both the stochasticity of the dynamics of genetic circuits as well as the partitioning schemes of molecules in cell division. The models that can be implemented can be used both to predict as well as to interpret results from measurements, thus providing a framework to investigate these key questions regarding cell longevity and adaptability to unpredictable environments.

REFERENCES

[1] A. Arkin, J. Ross, and H.H. Mcadams, “Stochastic Kinetic Analysis of Developmental Pathway Bifurcation in Phage lambda-Infected Escherichia coli Cells,” *Genetics*, vol. 149, 1998, pp. 1633-1648.

[2] H.H. Chang, M. Hemberg, M. Barahona, D.E. Ingber, and S. Huang, “Transcriptome-wide noise controls lineage choice in mammalian progenitor cells,” *Nature*, vol. 453, 2008, pp. 4-8.

[3] H. Maamar, A. Raj, and D. Dubnau, “Noise in Gene Expression Determines Cell Fate in Bacillus subtilis,” *Science*, vol. 526, 2008.

[4] G.M. Süel and M.B. Elowitz, “Tunability and Noise Dependence in Differentiation Dynamics,” *Science*, vol. 315, 2008.

[5] K.M. Herbert, A.L. Porta, B.J. Wong, R.A. Mooney, K. C, R. Landick, and S.M. Block, “Sequence-Resolved Detection of Pausing by Single RNA Polymerase Molecules,” *Cell*, vol. 125, 2006, pp. 1083-1094.

[6] K.F. Murphy and J.J. Collins, “Combinatorial promoter design for engineering noisy gene expression,” *PNAS*, vol. 104, 2007, pp. 12726-12731.

[7] D. Huh and J. Paulsson, “Random partitioning of molecules at cell division,” *PNAS*, vol. 108, 2011, pp. 15004-15009.

[8] A.S. Ribeiro, “Dynamics and evolution of stochastic bistable gene networks with sensing in fluctuating environments,” *Physical Review E*, vol. 78, 2008, pp. 1-9.

[9] R. Mcclure, “Mechanism and Control of Transcription Initiation in Prokaryotes,” *Annual Reviews Biochemistry*, vol. 54, 1985, pp. 171-204.

[10] R. Lutz, T. Lozinski, T. Ellinger, and H. Bujard, “Dissecting the functional program of Escherichia coli promoters : the combined mode of action of Lac repressor and AraC activator,” *Nucleic Acids Research*, vol. 29, 2001, pp. 3873-3881.

[11] D. Kennel and H. Riezman, “Transcription and translation initiation frequencies of the Escherichia coli lac operon,” *Journal of Molecular Biology*, vol. 14, 1977, pp. 1-21.

[12] H. Buc and W.R. Mccluret, “Kinetics of Open Complex Formation between Escherichia coli R N A Polymerase and the lac UV5 Promoter . Evidence for a Sequential Mechanism Involving Three Stepst,” *Biochemistry*, vol. 24, 1985, pp. 326-328.

[13] I. Golding, J. Paulsson, S.M. Zawilski, and E.C. Cox, “Real-Time Kinetics of Gene Activity in Individual Bacteria,” *Cell*, vol. 123, 2005, pp. 1025-1036.

[14] A.B. Lindner, R. Madden, A. Demarez, and E.J. Stewart, “Asymmetric segregation of protein aggregates is associated with cellular aging and rejuvenation,” *PNAS*, vol. 105, 2008.

[15] E.J. Stewart, R. Madden, and G. Paul, “Aging and Death in an Organism That Reproduces by Morphologically Symmetric Division,” *PLOS Biology*, vol. 3, 2005.

[16] R. Zhu, A.S. Ribeiro, D. Salahub, and S.A. Kauffman, “Studying genetic regulatory networks at the molecular level : Delayed reaction stochastic models,” *Journal of Theoretical Biology*, vol. 246, 2007, pp. 725-745.

[17] M.R. Roussel and R. Zhu, “Stochastic Kinetics Description of a Simple Transcription Model,” *Bulletin of Mathematical Biology*, vol. 68, 2006, pp. 1681-1713.

[18] A.S. Ribeiro, “Stochastic and delayed stochastic models of gene expression and regulation,” *Mathematical Biosciences*, vol. 223, 2010, pp. 1-11.

[19] M. Kærn, T.C. Elston, W.J. Blake, and J.J. Collins, “Stochasticity in Gene Expression: from Theories to Phenotypes,” *Nature Reviews*, vol. 6, 2005, pp. 451-464.

[20] D.T. Gillesple, “A General Method for Numerically Simulating the Stochastic Time Evolution of Coupled Chemical Reactions,” *Journal of Computational Physics*, vol. 434, 1976, pp. 403-434.

[21] M.R. Roussel and R. Zhu, “Validation of an algorithm for delay stochastic simulation of transcription and translation in prokaryotic gene expression,” *Physical Biology*, vol. 3, 2006, pp. 274-284.

[22] A. Spicher, O. Michel, M. Cieslak, J.-louis Giavitto, and P. Prusinkiewicz, “Stochastic P systems and the simulation of biochemical processes with dynamic compartments,” *Biosystems*, vol. 91, 2008, pp. 458-472.

[23] A.S. Ribeiro and J. Lloyd-price, “SGN Sim , a Stochastic Genetic Networks Simulator,” *Bioinformatics*, vol. 23, 2007, pp. 777-779.

[24] M. Kandhavelu, A. Häkkinen, O. Yli-Harja, and A.S. Ribeiro, “Single-molecule dynamics of transcription of the lar promoter,” *Physical Biology*, vol. 9, 2012.

[25] M. Kandhavelu, H. Mannerström, A. Gupta, A. Häkkinen, J. Lloyd-price, O. Yli-Harja, and A.S. Ribeiro, “In vivo kinetics of transcription initiation of the lar promoter in Escherichia coli . Evidence for a sequential mechanism with two rate-limiting steps,” *BMC Systems Biology*, vol. 5, 2011, p. 149.

[26] Y. Taniguchi, P.J. Choi, G.-wei 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, 2010.

[27] P.J. Choi, L. Cai, and X.S. Xie, “A Stochastic Single-Molecule Event Triggers Phenotype Switching of a Bacterial Cell,” *Science*, vol. 4142, 2008, pp. 442-446.

[28] J.A. Bernstein, A.B. Khodursky, P.-hsun 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,” *PNAS*, vol. 99, 2002, pp. 9697-9702.

[29] J.M. Pedraza and J. Paulsson, “Effects of Molecular Memory and Bursting on Fluctuations in Gene Expression,” *Science*, vol. 319, 2008.

[30] T. Rajala, A. Häkkinen, S. Healy, O. Yli-Harja, and A.S. Ribeiro, “Effects of Transcriptional Pausing on Gene Expression Dynamics,” *PLoS Computational Biology*, vol. 6, 2010, pp. 29-30.

[31] M.A. Sorensen, C.G. Kurland, and S. Pedersen, “Codon Usage Determines Translation Escherichia coli Rate in,” *Journal of Molecular Biology*, vol. 207, 1989, pp. 365-377.

[32] O. Yarchuk, J. Guillerez, and M. Dreyfus, “Interdependence of Translation , Transcription Degradation in the ZacZ Gene and mRNA,” *Journal of Molecular Biology*, vol. 226, 1992, pp. 581-596.

[33] D. Fusco, N. Accornero, B. Lavoie, S.M. Shenoy, J.-marie Blanchard, R.H. Singer, and E. Bertrand, “Single mRNA Molecules Demonstrate Probabilistic Movement in Living Mammalian Cells,” *Current Biology*, vol. 13, 2003, pp. 161-167.

[34] J. Paulsson, “Summing up the noise in gene networks,” *Nature*, vol. 427, 2004, pp. 415-418.

[35] J. Yu and X.S. Xie, “Probing Gene Expression in Live Cells, One Protein Molecule at a Time,” *Science*, vol. 1600, 2007.

[36] D.F. Browning, S.J.W. Busby, and C.S.J.W. B, “The regulation of Bacterial Transcription Initiation,” *Nature Reviews*, vol. 2, 2004, pp. 1-9.

[37] R.M. Saecker, M.T.R. Jr, and P. L, “Mechanism of Bacterial Transcription Initiation : Promoter Binding , Isomerization to Initiation-Competent Open Complexes , and Initiation of RNA Synthesis,” *Journal of Molecular Biology*, 2011, pp. 1-18.

[38] R. Lutz and H. Bujard, “Independent and tight regulation of transcriptional units in Escherichia coli via the LacR / O , the TetR / O and AraC / I 1 -I 2 regulatory elements,” *Nucleic Acids Research*, vol. 25, 1997, pp. 1203-1210.

[39] J. Peccoud and B. Ycart, “Markovian Modelling of Gene Product Synthesis,” *Theoretical Population Biology*, vol. 48, 1995, pp. 222-234.

[40] J. Mäkelä, J. Lloyd-price, O. Yli-Harja, and A.S. Ribeiro, “Stochastic sequence-level model of coupled transcription and translation in prokaryotes,” *BMC Bioinformatics*, vol. 12, 2011, p. 121.

[41] I. Golding and E.C. Cox, “RNA dynamics in live Escherichia coli cells,” *PNAS*, vol. 101, 2004, pp. 11310-11315.

[42] M. Lehnik-Habrink, H. Pförtner, L. Rempeters, N. Pietack, C. Herzberg, and J. Stülke, “The RNA degradosome in Bacillus subtilis : identification of CshA as the major RNA helicase in the multiprotein complex,” *Molecular Microbiology*, vol. 77, 2010, pp. 958-971.

[43] D. Huh and J. Paulsson, “Non-genetic heterogeneity from stochastic partitioning at cell division,” *Nature Genetics*, vol. 43, 2011.

[44] J. Lloyd-Price, M. Lehtivaara, M. Kandhavelu, S. Chowdhury, A.-B. Muthukrishnan, O. Yli-Harja, and A.S. Ribeiro, “Probabilistic RNA partitioning generates transient increases in the normalized variance of RNA numbers in synchronized populations of Escherichia coli,” *Molecular Biosystems*, vol. 8, 2012, pp. 565-571.

[45] D.A. McQuarrie, “Stochastic Approach to Chemical Kinetics,” *J. Appl. Probability*, vol. 4, 1967, p. 413.

[46] D.T. Gillespie, “Stochastic Simulation of Chemical Kinetics,” *Annual Review of Physical Chemistry*, vol. 58, 2007, pp. 35-55.

[47] D.T. Gillesple, “Exact Stochastic Simulation of Coupled Chemical Reactions,” *Journal of Physical Chemistry*, vol. 81, 1977, pp. 2340-2361.

[48] D.T. Gillesple, “A rigorous derivation of the chemical master equation,” *Physica A*, vol. 188, 1992, pp. 404-425.

[49] M.A. Gibson and J. Bruck, “Efficient Exact Stochastic Simulation of Chemical Systems with Many Species and Many Channels,” *Journal of Physical Chemistry*, vol. 104, 2000, pp. 1876-1889.

[50] Y. Cao, H. Li, L. Petzold, and J. Bruck, “Efficient formulation of the stochastic simulation algorithm for chemically reacting systems,” *Journal of Chemical Physics*, vol. 121, 2004.

[51] J.M. Mccollum, G.D. Peterson, C.D. Cox, M.L. Simpson, and N.F. Samatova, “The sorting direct method for stochastic simulation of biochemical systems with varying reaction execution behavior,” *Computational Biology and Chemistry*, vol. 30, 2006, pp. 39-49.

[52] H. Li and L. Petzold, “Logarithmic Direct Method for Discrete Stochastic Simulation of Chemically Reacting Systems,” 2006, pp. 1-11.

[53] R. Ramaswamy, N. González-segredo, and I.F. Sbalzarini, “A new class of highly efficient exact stochastic simulation algorithms for chemical reaction networks,” *Journal of Chemical Physics*, vol. 130, 2009.

[54] K.R. Schneider and T. Wilhelm, “Model reduction by extended quasi-steady-state,” *Journal of Mathematical Biology*, vol. 40, 2000, pp. 443-450.

[55] R.W. Hockney and J. it’ Eastwood, *Computer Simulation Using Particles*, 1988.

[56] A. Slepoy, A.P. Thompson, and S.J. Plimpton, “A constant-time kinetic Monte Carlo algorithm for simulation of large biochemical reaction networks,” *Journal of Chemical Physics*, vol. 128, 2008, pp. 1-8.

[57] T.-been Chen, H.H.-shing Lu, Y.-shien Lee, and H.-jen Lan, “Segmentation of cDNA microarray images by kernel density estimation,” *Journal of Biomedical Informatics*, vol. 41, 2008, pp. 1021-1027.

[58] L. Devroye, L. Györfi, and G. Lugosi, *A Probabilistic Theory of Pattern Recognition*, 1996.

[59] K. Adelman, A.L. Porta, T.J. Santangelo, J.T. Lis, J.W. Roberts, and M.D. Wang, “Single molecule analysis of RNA polymerase elongation reveals uniform kinetic behavior,” *PNAS*, vol. 99, 2002.

[60] M. Kandhavelu, J. Lloyd-price, A. Gupta, A.-B. Muthukrishnan, O. Yli-Harja, and A.S. Ribeiro, “Regulation of mean and noise of the in vivo kinetics of transcription under the control of the lac/ara-1 promoter,” *BMC Molecular Biology*, 2012, p. Submitted.

[61] P.M. Llopis, A.F. Jackson, O. Sliusarenko, I. Surovtsev, J. Heinritz, T. Emonet, and C. Jacobs-wagner, “Spatial organization of the flow of genetic information in bacteria,” *Nature*, vol. 466, 2010, pp. 77-81.

[62] Q. Wang, J. Niemi, C.-meng Tan, L. You, and M. West, “Image segmentation and dynamic lineage analysis in single-cell fluorescence microscopy,” 2009, pp. 1-16.

[63] R. Rosen, D. Biran, E. Gur, and E.Z. Ron, “Protein aggregation in Escherichia coli : role of proteases,” *FEMS Microbiology Letters*, vol. 207, 2002, pp. 9-12.

[64] R. Ierusalimschy and W. Celes, *The Evolution of Lua*.

[65] G.S. Gordon, D. Sitnikov, C.D. Webb, A. Teleman, A. Straight, R. Losick, A.W. Murray, and A. Wright, “Chromosome and Low Copy Plasmid Segregation in E . coli : Visual Evidence for Distinct Mechanisms,” *Cell*, vol. 90, 1997, pp. 1113-1121.

[66] A. Ribeiro, R.U.I. Zhu, and S.A. Kauffman, “A General Modeling Strategy for Gene Regulatory Networks with Stochastic Dynamics,” *Journal of Computational Biology*, vol. 13, 2006, pp. 1630-1639.

[67] B.P. Cormack, R.H. Valdivia, and S. Falkow, “FACS-optimized mutants of the green fluorescent protein (GFP) (,” *Gene*, vol. 173, 1996, pp. 33-38.

[68] A.S. Ribeiro and S.A. Kauffman, “Noisy attractors and ergodic sets in models of gene regulatory networks,” *Journal of Theoretical Biology*, vol. 247, 2007, pp. 743-755.

[69] J. Blakes, J. Twycross, F.J. Romero-campero, and N. Krasnogor, “The Infobiotics Workbench : an integrated in silico modelling platform for Systems and Synthetic Biology,” *Bioinformatics*, vol. 27, 2011, pp. 3323-3324.

[70] J. Twycross, M. Bennett, and N. Krasnogor, “A High-Performance Multicompartment Stochastic Simulator for Executable Biology.”