COMPUTATIONAL INVESTIGATIONS OF EVOLUTIONARY TRANSITIONS DURING DEVELOPMENT OF THE CELLULAR TRANSLATION AND TRANSCRIPTION MACHINERY

BY

ELIJAH ROBERTS

DISSERTATION

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Biophysics & Computational Biology in the Graduate College of the University of Illinois at Urbana-Champaign, 2010

Urbana, Illinois

Doctoral Committee:

Professor Zaida (Zan) Luthey-Schulten, Chair
Professor Taekjip Ha
Professor Wen-mei W Hwu
Assistant Professor Nathan D Price
Computational Investigations of Evolutionary Transitions during Development of the Cellular Translation and Transcription Machinery
Roberts, Elijah. 2010.

Abstract
Evolutionary transitions, times at which the behavior of evolution as a dynamic system dramatically changes, have occurred many times throughout the history of life on Earth. Carl Woese proposed that one such transition occurred at the root of the universal phylogenetic tree as life crossed a “Darwinian threshold”. He theorized that evolution before the transition was communal, involving massive horizontal transfer of genes, whereas evolution afterward followed a more vertical path, similar to that observed today. Christian de Duve, under the term “singularities”, similarly proposed a series of such transformational events in the history of life, including the development of a compartmentalized cellular nucleus. The work presented in this dissertation describes a series of computational studies designed to explore two of these transitions: the divergence of the cellular translation machinery in the three organismal lineages and the development of strategies for coping with the effects of spatial heterogeneity on gene regulation. Several new computational methodologies developed to address these questions are also presented.

Ribosomal signatures, idiosyncrasies in the ribosomal RNA and/or proteins, are characteristic of the individual domains of life. Contributions from these signatures represent a significant fraction of the phylogenetic signal separating the three domains of life. The evolutionary origin of the signatures is analyzed and discussed, with the likely explanation being horizontal gene transfer within each organismal lineage following its divergence from the ancestral pool. Additional support for this hypothesis comes from a study of the phylogeny of the universal ribosomal proteins in Bacteria, where the large number of available genomes can help to decompose the complex history of these proteins.

Transcription networks control the phenotype of modern cells, regulating the expression of proteins according to a genetic program. Bacteria and archaea couple transcription and translation in the cytoplasm, where the processes are subject to a great deal of spatial heterogeneity and the effects of the in vivo environment. Eukarya, on the other hand, have segregated transcription into a controlled compartment via the evolution of the nucleus. To understand the effect an evolutionary transition to complete segregation would have had, the effects of spatial heterogeneity are studied in a simple bacterial network, namely the regulatory network encoded in the lac operon. A novel method is presented for studying the effect of incorporating spatial information and molecular crowding into stochastic models of genetic circuits. By comparing to the well-stirred model, it is shown that spatial degrees of freedom and in vivo crowding can change both the noise and the mean behavior of a circuit. The spatial noise is a component of the extrinsic noise of a genetic system and bounds are placed on its contribution.

Evolutionary transitions leave distinct signatures in the fabric of the cell. By studying these “molecular fossils” one can recover physical details about the transitions themselves as well as about the overall dynamics of the evolutionary process.
To my son Jacob, for changing my life.
Acknowledgements

First and foremost I would like to thank my wife Jaime and my son Jacob, both of whom I first met during my time in Illinois working on this dissertation. Their support and company give my life its meaning and keep me coming home at night and waking up in the morning. Of course I owe a special amount of thanks to my parents, Peggy and Lee, for their continuous support and for instilling me with curiosity since before I can remember. I would clearly not be me without their unique pattern and perspective on life. Likewise, my grandparents John and Helen Burnett, both of whom passed away during my years here, were a large formative factor in making me the person I am.

I want to thank all of the members of the Luthey-Schulten group and other scientists at the University of Illinois for making this the most scientifically exciting place for graduate work one could imagine. A few people who worked with me on various projects in no particular order are Patrick O’Donoghue, Taras Pogorelov, Anurag Sethi, John Eargle, Jonathan Montoya, Dan Wright, Ke Chen, Leonardo Sepulveda, Andrew Magis, and John Stone.

When I first came to Illinois, I was totally uneducated about biology. I want to acknowledge a deep debt of gratitude to Carl Woese for trying to teach me the proper way to think about biology and evolution. Of course it is impossible to learn everything Carl has to teach about evolution in so short a time, but I hope I have at least learned enough to keep asking interesting questions for a long time to come.

I would like to thank my committee members Wen-mei Hwu, Nathan Price, and TJ Ha for providing me with their insight and expertise in biology, computer science, and physics. The overlap of these three fields is, in my opinion, the most exciting area of science today. I would like to thank Ido Golding as well for many insightful discussions even though he was unable to sit on my committee.

Finally, I want to sincerely thank Zan for teaching me things so numerous it would be impossible to recall them all. During my time in her group she taught me how to be a scientist, how to perform research, and perhaps most importantly, how to ask interesting questions. In particular, Zan’s insight on both evolution and physical processes out of equilibrium and how to apply non-equilibrium techniques to biological systems has really inspired me. Zan’s willingness to work with a curious, computational minded person who was essentially starting devoid of biological knowledge and with only a limited background in physical methods is what made my progress possible.

Needless to say without the support of the scientific funding agencies, much research in this country would come to a halt. During the course of my Ph.D. work I have been funded by numerous agencies. The NSF, DOE and NIH all were the source of support at various times. Specific grants are acknowledged in the appropriate publications.
# Table of Contents

List of Tables  .................................................................................................................. viii

List of Figures  .................................................................................................................... ix

List of Abbreviations  .......................................................................................................... xi

Chapter 1  Introduction  ...................................................................................................... 1

Chapter 2  Molecular Signatures of Ribosomal Evolution  .............................................. 4
  2.1  Background  ................................................................................................................ 4
  2.2  Ribosomal Signatures  ............................................................................................... 5
    2.2.1  Evolution of rRNA signatures  .............................................................................. 5
    2.2.2  Domain specific ribosomal proteins as signatures  .............................................. 9
    2.2.3  Signatures in genomic organization  .................................................................. 10
  2.3  Signatures as “Molecular Fossils”  ........................................................................ 11
    2.3.1  Correlation of signatures in rRNA and ribosomal proteins  ............................. 11
    2.3.2  Evolution of ribosomal proteins L36 and L40e  ................................................ 13
    2.3.3  Final remarks  .................................................................................................... 15
  2.4  Methods  ..................................................................................................................... 15
    2.4.1  Creation of evolutionary profiles  ....................................................................... 15
    2.4.2  Identification of sequence and structural signatures  ........................................ 16
    2.4.3  Sequence phylogenetic analysis of 16S and 23S rRNAs  ................................. 16
    2.4.4  Structural phylogenetic analysis of 23S rRNA  .................................................. 16
    2.4.5  Mutual information analysis of S4 and 16S rRNA coevolution  ...................... 17
    2.4.6  Molecular dynamics simulation of r-protein L40e in the archaeal LSU  ........... 17

Chapter 3  Communal Sharing of Ribosomal Proteins among the Early Bacteria  .......... 19
  3.1  Background  ............................................................................................................... 19
  3.2  Classification and Phylogeny of Ribosomal Protein S4  ......................................... 21
    3.2.1  Sequence alignment and classification of bacterial r-protein S4  .................... 21
    3.2.2  Phylogenetic reconstruction of S4’s evolutionary history  ............................... 25
  3.3  Horizontal Gene Transfer of S4  ............................................................................. 27
    3.3.1  Identification of paralogous and duplicated S4 genes  ................................... 27
List of Tables

2.1 Structural signatures between archaeal and bacterial 16S rRNA . . . . . . . . . . . . . . . . 7
2.2 Structural signatures between archaeal and bacterial 23S rRNA . . . . . . . . . . . . . . . . 8
2.3 Percentage of domain specific ribosomal proteins in the ribosomal subunits . . . . . . . . 9

3.1 Genomes containing multiple copies of the S4 gene . . . . . . . . . . . . . . . . . . . . . . 28
3.2 Taxonomic distributions of the C+/− ribosomal proteins . . . . . . . . . . . . . . . . . . . . 36

5.1 Available multiparticle diffusion kernels . . . . . . . . . . . . . . . . . . . . . . . . . . . 76
5.2 Whole cell GPU performance . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 77
5.3 GPU memory required according to lattice size . . . . . . . . . . . . . . . . . . . . . . . . . 78
5.4 Whole cell GPU performance . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 90
5.5 Probability calculation performance on GPU . . . . . . . . . . . . . . . . . . . . . . . . . 93

6.1 Reactions and rate constants used in the stochastic model of the lac circuit . . . . . . . . 98
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Phylogenetic tree of the 23S rRNA</td>
<td>5</td>
</tr>
<tr>
<td>2.2</td>
<td>Sequence and structural signatures in the rRNA</td>
<td>6</td>
</tr>
<tr>
<td>2.3</td>
<td>Contributions of signatures to domain phylogeneies</td>
<td>7</td>
</tr>
<tr>
<td>2.4</td>
<td>Bacterial and archaean signatures in environmental sequences</td>
<td>9</td>
</tr>
<tr>
<td>2.5</td>
<td>Spatial analogs in the LSU</td>
<td>10</td>
</tr>
<tr>
<td>2.6</td>
<td>Genomic context of ribosomal protein clusters</td>
<td>11</td>
</tr>
<tr>
<td>2.7</td>
<td>Signature interactions between r-protein S4 and the SSU</td>
<td>12</td>
</tr>
<tr>
<td>2.8</td>
<td>Model of L40e in the archaeal LSU</td>
<td>14</td>
</tr>
<tr>
<td>3.1</td>
<td>Multiple sequence alignment of ribosomal protein S4</td>
<td>22</td>
</tr>
<tr>
<td>3.2</td>
<td>Consensus phylogenetic tree of S4 in <em>Proteobacteria</em></td>
<td>24</td>
</tr>
<tr>
<td>3.3</td>
<td>Consensus unrooted phylogenetic tree of S4</td>
<td>26</td>
</tr>
<tr>
<td>3.4</td>
<td>Expansion of <em>Clostridia</em> branches of the consensus S4 phylogenetic tree</td>
<td>29</td>
</tr>
<tr>
<td>3.5</td>
<td>Expansion of <em>Actinobacteria</em> branches of the consensus S4 phylogenetic tree</td>
<td>30</td>
</tr>
<tr>
<td>3.6</td>
<td>Genomic content near S4 and the α-operon</td>
<td>32</td>
</tr>
<tr>
<td>3.7</td>
<td>Occurrence of different types of S4 throughout the bacterial phylogeny</td>
<td>33</td>
</tr>
<tr>
<td>3.8</td>
<td>Possible evolutionary pathways of S4</td>
<td>35</td>
</tr>
<tr>
<td>4.1</td>
<td>Overview of the MultiSeq environment</td>
<td>43</td>
</tr>
<tr>
<td>4.2</td>
<td>BLAST search results viewer</td>
<td>44</td>
</tr>
<tr>
<td>4.3</td>
<td>Grouping in MultiSeq</td>
<td>45</td>
</tr>
<tr>
<td>4.4</td>
<td>MultiSeq tools</td>
<td>46</td>
</tr>
<tr>
<td>4.5</td>
<td>Comparison of phylogenies with different degrees of imbalance</td>
<td>55</td>
</tr>
<tr>
<td>4.6</td>
<td>Orthogonal encoding of a protein sequence alignment</td>
<td>57</td>
</tr>
<tr>
<td>4.7</td>
<td>QR cutoff analysis</td>
<td>58</td>
</tr>
<tr>
<td>4.8</td>
<td>Mean imbalance of evolutionary profiles as a function of profile size</td>
<td>59</td>
</tr>
<tr>
<td>4.9</td>
<td>Imbalance statistics for phylogenetic trees of maximum-entropy evolutionary profiles</td>
<td>59</td>
</tr>
<tr>
<td>4.10</td>
<td>Distribution of the percent identity cutoffs in maximum-entropy evolutionary profiles</td>
<td>60</td>
</tr>
<tr>
<td>4.11</td>
<td>Mean imbalance for nucleic acid maximum-entropy evolutionary profiles</td>
<td>61</td>
</tr>
<tr>
<td>4.12</td>
<td>Comparison of QR and phylogenetic diversity</td>
<td>62</td>
</tr>
<tr>
<td>4.13</td>
<td>Covariation analysis of 16S rRNA and S4</td>
<td>64</td>
</tr>
<tr>
<td>5.1</td>
<td>Particle move probabilities</td>
<td>71</td>
</tr>
</tbody>
</table>
List of Abbreviations

CA – cellular automata
CME – chemical master equation
CPU – central processing unit
CUDA – Compute Unified Device Architecture
DGL – differential gene loss
ds-protein – domain specific ribosomal protein
EF-Tu – elongation factor Tu
EP – evolutionary profile
ERM – equal-rates Markov
GFP – green fluorescent protein
GPU – graphics processing unit
HGT – horizontal gene transfer
indel – insertion or deletion
IPTG – isopropyl β-D-1-thiogalactopyranoside
LacI – lac repressor
LacY – lactose permease
LSU – large subunit
ME-EP – maximum-entropy evolutionary profile
ML – maximum-likelihood
mRNA – messenger RNA
MSA – multiple sequence alignment
MSD – mean square displacement
nm – nanometers
ns – nanoseconds
PDE – partial differential equation
RBS – ribosomal binding site
RNAP – RNA polymerase
RDME – reaction-diffusion master equation
rRNA – ribosomal RNA
r-protein – ribosomal protein
SSU – small subunit
SI – supporting information
SRP – signal recognition particle
TMG – thiomethyl-β-D-galactoside
TOC – table of contents
UPT – universal phylogenetic tree
A. metalliredigens – Alkaliphilus metalliredigens
A. oremlandii – Alkaliphilus oremlandii
B. subtilis – Bacillus subtilis
C. acetobutylicum – Clostridium acetobutylicum
C. novyi – Clostridium novyi
D. radiodurans – Deinococcus radiodurans
E. coli – Escherichia coli
F. magna – Finegoldia magna
H. marismortui – Haloarcula marismortui
L. borgpetersenii – Leptospira borgpetersenii
M. flagellatus – Methylcobacillus flagellatus
M. tuberculosis – Mycobacterium tuberculosis
P. ingrahamii – Psychromonas ingrahamii
S. arenicola – Salinispora arenicola
S. coelicolor – Streptomyces coelicolor
S. solfataricus – Sulfolobus solfataricus
S. tropica – Salinispora tropica
T. thermophilus – Thermus thermophilus
Chapter 1

Introduction

“Evolution is the study of the biology of becoming not the biology of the here-and-now.” – Carl Woese

Evolutionary transitions, times at which the behavior of evolution as a dynamic system dramatically changes, have occurred many times throughout the history of life on Earth. Carl Woese proposed that one such transition occurred at the root of the universal phylogenetic tree as life crossed a “Darwinian threshold” [1]. He theorized that evolution before the transition was communal, involving massive horizontal transfer of genes, whereas evolution afterward followed a more vertical path, similar to that observed today. Christian de Duve, under the term “singularities”, similarly proposed a series of such transformational events in the history of life [2], including the development of a compartmentalized cellular nucleus spatially segregating transcription from translation. Evolutionary transitions leave distinct signatures in the fabric of the cell. By studying these “molecular fossils” one can recover physical details about the transitions themselves as well as about the overall dynamics of the evolutionary process.

This dissertation describes a series of computational studies designed to explore two evolutionary transitions: the divergence of the cellular translation machinery in the three organismal lineages and the development of strategies for coping with the effects of spatial heterogeneity on gene regulation. It is grouped into two main topic areas. The first three chapters deal with the divergence of the cellular translation machinery, namely the ribosome and the ribosomal proteins, from the perspective of evolutionary biology. The translation machinery is one of the core information processing systems of the cell, so understanding its history is vital to understanding the history of cellular life. Also presented is an overview of MultiSeq, our widely used bioinformatics analysis software, which was developed specifically to facilitate the work presented.

Chapter 2 – Ribosomal signatures, idiosyncrasies in the ribosomal RNA (rRNA) and/or proteins, are characteristic of the individual domains of life. As such, insight into the early evolution of the domains can be gained from a comparative analysis of their respective signatures in the translational apparatus. Here, we identify signatures in both the sequence and structure of the rRNA and analyze their contributions to the universal phylogenetic tree. We find correlations between the rRNA signatures and signatures in the ribosomal proteins showing that the rRNA signatures coevolved with both domain-specific and universal ribosomal proteins. Finally, we show that the genomic organization of the universal ribosomal components contains these signatures as well. From these studies, we propose the ribosomal signatures are remnants
of an evolutionary-phase transition that occurred as the cell lineages began to coalesce and so should be reflected in corresponding signatures throughout the fabric of the cell and its genome.

**Chapter 3** – The universal ribosomal protein S4 is essential for the initiation of small subunit ribosomal assembly and translational accuracy. Being part of the information processing machinery of the cell, the gene for S4 is generally thought of as being inherited vertically and has been used in concatenated gene phylogenies. Here we report the evolution of ribosomal protein S4 in relation to a broad sharing of zinc/non-zinc forms of the gene and study the scope of horizontal gene transfer (HGT) of S4 during bacterial evolution. The complex history presented for “core” protein S4 suggests the existence of a gene pool before the emergence of bacterial lineages and reflects the pervasive nature of HGT in subsequent bacterial evolution. This has implications for both theoretical models of evolution and practical applications of phylogenetic reconstruction as well as the control of zinc economy in bacterial cells.

**Chapter 4** – Since the publication of the first draft of the human genome in 2000, bioinformatics data have been accumulating at an overwhelming pace. Currently, millions of sequences, including more than a thousand complete genomes, and tens of thousands of structures of proteins and nucleic acids are available in public databases. Finding correlations in and between these data to answer critical research questions is extremely challenging. Here we present MultiSeq, a unified bioinformatics analysis environment that allows one to organize, display, align and analyze sequence, structure, phylogenetic, and genomic data for proteins and nucleic acids. While special emphasis is placed on analyzing the data within the framework of evolutionary biology, the environment is also flexible enough to accommodate other usage patterns. We also present a unique new method for selecting a representative set from available sequence data, termed an “evolutionary profile”, by maximizing the Shannon entropy of the profile. Tools such as the ones presented are critical for working with the large data sets emerging in the post-genomic era.

The final two chapters represent the beginnings of a new method to analyze the evolution of cellular transcriptional regulatory networks in the context of physical biology. The work presented represents only the first step of such an approach, namely the development and application of computational techniques to investigate cellular regulation while accounting for spatial degrees of freedom. The ultimate question of interest is how cells evolved to deal with increasingly complex spatial heterogeneity while maintaining their regulatory programming.

**Chapter 5** – One of the long-term goals of computational biology is the development of a model for an entire cell under natural conditions. Recent advances in the experimental localization of cellular components have catalyzed interest in the theoretical and computational challenges associated with such a model. To address the problem of performing long time simulations of biochemical pathways under *in vivo* cellular conditions, we have developed a lattice-based,
reaction-diffusion model that uses the graphics processing unit (GPU) as a computational co-processor. In this study we present our three-dimensional model for in vivo reaction-diffusion that exploits the calculation capabilities of the GPU. We then compare results from free and in vivo diffusion simulations to the anomalous behavior expected from molecular crowding. Based on the comparison and the results of performance testing, we conclude that the lattice-based model, while an approximation to reality, shows great potential for studying cellular processes where dynamics on the time-scale of the cell cycle are important.

Chapter 6 – Deterministic and stochastic models used to explore the dynamics of cellular biochemical networks typically ignore spatial degrees of freedom by assuming the cell is well-stirred. Spatial heterogeneity has been neglected due to the lack of both data regarding cellular localization and computational methodologies to simulate such models. Advances in in vivo imaging techniques, including cryo-electron tomography and single-molecule fluorescence microscopy, have begun to reveal the organization and dynamics of biomolecules inside the cell. Likewise, GPUs now provide the computational power to perform three-dimensional simulations of cell-scale models. Here, the effects of incorporating spatial information and molecular crowding into a stochastic model of the lactose utilization genetic circuit are reported. We use our recently developed lattice-based Monte Carlo simulation technique to sample the reaction-diffusion master equation describing the lac circuit in an Escherichia coli cell. Parameters are obtained from published in vivo single molecule studies. By comparing to the well-stirred model, it is shown that spatial degrees of freedom and in vivo crowding can change both the noise and the mean behavior of a circuit. Such spatial noise is a component of the extrinsic noise of a genetic system and we put bounds on its contribution. Finally, the model suggests new single molecule experiments to probe the lac circuit and provides estimates of the spatial and temporal resolution required to perform them.
Chapter 2

Molecular Signatures of Ribosomal Evolution

2.1 Background

A huge and exponentially increasing data set regarding the molecular makeup of cells has accumulated over
the last several decades. Biologists today routinely ask questions of the data that are far more deeply probing
than was previously possible. What is not generally appreciated, however, is that large data sets of this type
tend to bring into question the conceptual framework within which the questions themselves are posed.
An especially informative example is our understanding of the cellular translation mechanism. In the past,
the mechanism was conceptualized and probed in a reductionist, “particle” framework while understanding
today comes increasingly from multimodal analyses. The questions and answers bespeak a highly integrated
mechanism, whose essence would seem to lie in its delocalized, collective properties.

This perceptual change obviously applies not only to translation, but embraces all biological organiza-
tion, all things biological. Ultimate explanations in biology will come largely in terms of processes – a
process perspective that unavoidably leads back to the dynamics of evolution, the process that gives rise to
all the subordinate biological processes constituting what we take to be biology today. The process of evo-
lution is a forteriori non-uniform and while its sporadic nature can be glimpsed throughout the fabric of the
cell, perhaps its clearest markings are seen in the signatures of the translation apparatus, i.e., the ribosome
and its translation factors.

Evidence today strongly suggests that a highly developed translation system was a necessary condition
for the emergence of cells, as we know them [1]. In the universal phylogenetic tree (UPT) format this
maturation of the translation system seems to be represented by the tree’s basal branchings, where first the
bacterial and then the archaeal and eukaryotic lineages appear individually to emerge. What lies beneath
this “root” locus, the evolution leading up to it, cannot be captured in familiar tree representation. It would
seem to be some distributed universal ancestral state from which the (three) primary organismal lineages
materialized via one or a brief series of major evolutionary saltations in which the state of the evolving
cellular organization and the accompanying evolutionary dynamic underwent dramatic change. The aborig-

---

inal evolutionary dynamic may have been “Lamarckian” in the sense that it seems likely to have involved massive, pervasive horizontal transfer of genes (HGT) – innovation sharing. The kind and frequency of the HGT envisioned would make evolution early on effectively communal. This communal evolutionary dynamic comes to an end relatively suddenly and transforms largely into the familiar genealogical dynamic when various organismal elements in the community reach critical stages wherein their organizations drastically change – which leads to refinement and “individualization”. These we call Darwinian transitions [1]. Certain signatures in the ribosome, i.e., idiosyncrasies in its RNA (rRNA) [3–6] and/or proteins (r-proteins) characteristic of the individual domains of life were locked in place at this time – becoming molecular fossils that are telling of the phase transitions.

The availability of genomic data and crystal structures for the bacterial small subunit (SSU) and the bacterial and archaeal large subunit (LSU) allows us now to extend the previous analyses of the ribosomal signatures both in depth, by including the r-proteins, and in scope, by looking at signatures at the levels of structure and genomic organization. Using a variety of techniques, we herein investigate the evolution of the molecular signatures of translation. Understanding the characteristics of that process will help us gain insight into the early evolution of translation, and therefore, of early cellular life.

2.2 Ribosomal Signatures

2.2.1 Evolution of rRNA signatures

The 16S rRNA has become the molecular standard in studying evolutionary relationships between organisms [7]. However, the 23S rRNA has followed a very similar (if not identical) evolutionary path, as shown by the congruence of its sequence phylogeny with the UPT (Figure 2.1). The 23S rRNA therefore provides

![Figure 2.1: Sequence phylogenetic tree of the 23S rRNA. Archaea are shown in blue and bacteria in red. H. halobium, S. elongatus, and S. sp were removed from the tree due to low support values. The tree was reconstructed using MrBayes and drawn using the program Unrooted. (Figure courtesy of J Montoya)](image-url)
additional, complimentary data that can be tapped to study the evolution of the ribosome.

The 16S and 23S rRNAs each have a high degree of sequence identity, with 30-40% of the well-aligned positions between bacteria and archaea being conserved. Yet despite this large degree of identity, there are significant phylogenetic signals in the pattern of change of the remaining nucleotides that can reveal the evolutionary history of the molecules. Among the strongest signals are the signatures, the regions that are constant and unique to, i.e., characteristic of, a particular domain of life. There appear to be two general kinds of signatures here. Sequence signatures comprise positions in the primary structure whose compositions remain constant in one domain of life but occur rarely in the other domains. Structural signatures are regions in the secondary and/or tertiary structure that have a unique configuration in a given domain.

We identified 69 distinguishing sequence signatures between the bacterial and archaeal 16S rRNAs using the sequences of 2,735 organisms and 119 such between their 23S rRNAs based upon 441 sequences (Figure 2.2). The sequence signatures constitute ~5% of the nucleotides in each molecule. Logically, since the compositions of these positions are conserved across the entirety of and unique to one given domain, their idiosyncrasy must have evolved in the ancestral stem of that domain. Therefore, they should have a large impact on the measure of the phylogenetic separation of the two domains; i.e., the distance between the roots of the bacterial and archaeal sub-branches.

To estimate the contribution of the signatures to the overall phylogenetic signal, we performed a phylogenetic analysis of the 16S and 23S rRNAs both with and without the sequence signatures. Figures 2.3(a) and (b) show the effect of removing from the calculation the 5% of the sequence positions that constitute the characteristic sequence signatures: a 42% decrease in the separation between the bacterial and archaeal sub-branches for the 16S rRNA and a 28% decrease for the 23S rRNA. The decrease in separation holds for a range of signature cutoffs from strict to lax conservation (95% – 80%). There is no appreciable change in the branchings or distances within the archaeal and bacterial sub-trees; the sequence signatures carry only information distinguishing the two domains of life. With such a strong signal, it is not surprising that the three domains of life could be identified and distinguished in 1977 using only oligonucleotides created by T1 ribonuclease cleavage of 16S rRNA [8; 9].

In addition to the sequence signatures in the primary structure, there are regions of the rRNA that contain structural signatures in the secondary or tertiary structure. Such structural signatures can be of three types: (i) insertions or deletions (indels) that are characteristically present in one domain of life but absent in another, (ii) regions of the rRNA in which the secondary (and therefore tertiary) structure differs between

![Figure 2.2: Location of sequence signatures (red) and structural signatures (blue) in the aligned 16S and 23S rRNA from two bacteria and one archaea.](image-url)
two domains, or (iii) regions that are similar in secondary structure but differ in their tertiary conformation.

Using a combination of sequence- and structure-based techniques, we identified 6 structural signatures distinguishing the bacterial and archaeal 16S rRNAs and 14 distinguishing their 23S rRNAs (Tables 2.1 and 2.2). Since a crystal structure of the archaeal SSU has not been solved, structural signatures for the 16S rRNA are limited to types (i) and (ii) and are only reliable because of the high quality of the available 16S rRNA alignments [6]. To exclude crystallization artifacts, type (iii) structural signatures in the 23S rRNA were included only if supported by specific interactions with r-proteins.

The contribution of the structural signatures to the separation between the bacteria and the archaea was calculated as above, by excluding regions containing the structural signature during a sequence phylogenetic

Table 2.1: Structural signatures between archaeal and bacterial 16S rRNA

<table>
<thead>
<tr>
<th>rRNA Domain</th>
<th>Domain</th>
<th>Nucleotides</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>E. coli</td>
<td>T. thermophilus</td>
<td>H. marismortui</td>
<td></td>
</tr>
<tr>
<td>3. II</td>
<td>511,540</td>
<td>494,523</td>
<td>442,449,478,479</td>
</tr>
</tbody>
</table>
analysis. The change in the separation between the bacterial and archaeal sub-branches was significantly less than for the sequence signatures (8% for the 16S, 16% for the 23S).

Structural phylogenetic methods [10; 11], which include a framework for modeling indels, provide a (possibly more reliable) alternative for evaluating the phylogenetic contribution of the structural signatures. A structure-based phylogenetic tree (Figure 2.3(c)) was generated using the 23S rRNA structures of one archaeal example and three bacterial: *Haloarcula marismortui* [12] and then *Deinococcus radiodurans* [13], *Escherichia coli* [14], and *Thermus thermophilus* [15]. It reveals a deep separation between the archaeal and the bacterial 23S rRNA structures, similar to that seen in sequence-based phylogenetic trees. Removing the structural signatures from the structural phylogenetic analysis reduces the separation between the two domains by 50%. The sequence signatures make no contribution to the separation in the structural phylogeny, since the signature nucleotides (despite having different identities) occupy homologous positions in the overall structure. This structural phylogenetic analysis leads us to conclude that the structural signatures are as important as the sequence signatures in defining the differences between the domains of life.

One of the primary indications that the RNA signatures are, in fact, remnants of an evolutionary saltation is their discrete character. There is no signature continuum between the domains of life; organisms either have the bacterial, the archaeal, or the eukaryal character, with a sizeable two-domain signature that links the archaeal and eukaryal domains [16]. We have checked for the presence of the archaeal and bacterial 16S rRNA sequence signatures in >90,000 environmental sequences (see Figure 2.4) from the Greengenes database [17]. These sequences represent a much wider sampling from the organismal pool than the cultured
Figure 2.4: Distribution of the presence of bacterial and archaeal signatures in 90,000 environmental 16S rRNA sequences used initially to identify the signatures. Again, no exceptions are seen; no “gray area” exists between the archaeal and bacterial signatures: the ribosome is either of bacterial or archaeal nature.

### 2.2.2 Domain specific ribosomal proteins as signatures

Comparative analysis of the available sequence and structure data allows us to infer whether a protein existed in the gene pool before the divergence of the primary organismal lineages. The universally distributed r-proteins exhibit what is called the canonical pattern (as defined by Woese et al. [16]), wherein the various taxa group into three distinct clusters (bacteria, archaea, eukarya) with the latter two showing the most structure and sequence similarity. While the canonical pattern provides evidence that the universal r-proteins were present at the so-called base of the UPT, the situation is less clear with regard to the remaining ribosomal proteins.

It is well-known that approximately half of all the r-proteins are confined to a subset of the domains of life (domain specific ribosomal proteins (ds-proteins); see Table 2.3). Practically all of the archaeal but none of the bacterial ds-proteins are present in eukarya, consistent with the notion that the bacterial lineage diverged from some ancestral “stem” before either the archaeal or eukaryal lineages. Since the presence of these ds-proteins within their specific domain(s) of life is conserved (with a few exceptions [18]) their existence represents another of the signatures distinguishing the ribosome between the domains. The evolutionary history of the ds-proteins can therefore be informative as to the history of the signatures in general.

Further characteristic signatures provided by the ds-proteins can be seen in their binding locations. A structural superposition of the LSUs from *T. thermophilus* and *H. marismortui* establishes six pairs of spatial analogues, ds-proteins that have no detectable structure or sequence homology but interact with the

<table>
<thead>
<tr>
<th></th>
<th>SSU</th>
<th>LSU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bacteria</td>
<td>8/23 (35%)</td>
<td>14/34 (41%)</td>
</tr>
<tr>
<td>Archaea</td>
<td>13/28 (46%)</td>
<td>20/40 (50%)</td>
</tr>
<tr>
<td>Eukarya</td>
<td>17/32 (53%)</td>
<td>26/46 (56%)</td>
</tr>
</tbody>
</table>
same region of the rRNA [12; 19] (Figure 2.5). Despite the lack of homology, spatial analogs often form similar contacts with the RNA. The overlap of the LSUs also reveals several cases in which ds-proteins superimpose on structural signatures in the 23S rRNA or on r-protein extensions in the other domain of life. While the phylogenetic contribution of the ds-proteins to the separation between the domains cannot be strictly calculated, they absolutely give the ribosome a distinct structural character in each domain.

2.2.3 Signatures in genomic organization

A well-documented trait of the universal r-proteins is clustering of their genes in a genome. In many bacteria, all of the universal r-protein genes (except that of S15) are grouped into a few conserved genomic clusters along with the genes of other universally distributed proteins involved in the translation and transcription processes. Likewise, in many archaea the universal r-protein genes (except those of S15 and L16) are organized into similar groups (see Figure 2.6). We have analyzed these genomic clusters in representative bacterial and archaeal genomes looking for characteristic domain specific differences between them.

The relative ordering of universal r-protein genes within a gene cluster is very similar between the bacterial and archaeal genomes, showing just two signatures. First, the order of the genes of r-proteins S4 and S11 is reversed between the two domains [20], and second, the gene for r-protein L16 is missing from the corresponding gene cluster in archaea [21]. Despite the conservation in relative ordering of the universal r-protein genes with a cluster, the clusters themselves are organized differently in the two domains of life. The boundaries dividing the clusters themselves are located near different genes in each domain.

A majority of the genes of the ds-proteins are distributed as either isolated genes or in domain specific clusters. Exceptions are the genes of r-proteins L36, L17, and L33 in bacteria, and L30e, S4e, L32e, L19e, and L18e in archaea. Interestingly, these eight ds-protein genes are all located in the clusters containing the universal r-protein genes. The position of each ds-protein gene within a cluster is conserved within the domain of life and its presence does not perturb the ordering of nearby universal r-protein genes. These ds-protein genes can be considered structural signatures of the bacterial and archaeal genomes. Two of the
three bacterial specific r-proteins whose genes are located in these clusters (L17 and L33) are known to have spatial analogs in the archaeal LSU, and L36 may have as well (see below).

2.3 Signatures as “Molecular Fossils”

2.3.1 Correlation of signatures in rRNA and ribosomal proteins

Conservation of the signatures within a given domain indicates that they are functionally important, and it is also clear from their phylogenetic distribution that they must have evolved shortly after the divergence of the cellular lineages. Correlations between sequence, structure, and genomic signatures of the ribosome offer insight into their functional relationships and help to answer questions about the ribosome’s evolution.

Both evolutionary and dynamical correlations result from direct physical contact between signatures. Roughly half of the domain specific LSU r-proteins and nearly all of the 23S rRNA structural signatures interact with each other. In each interaction, a ds-protein and an rRNA structural signature create a domain specific connection between distant regions of the 23S rRNA sequence. Expansion of the network of interactions within the ribosome in this manner is a well-known theme in the evolution of the ribosome following the divergence of the lineages [12].

Some interactions between the ds-proteins and the rRNA structural signatures do not expand the interaction network, but instead reconnect it in a different pattern. There are large differences in the tertiary structure of helices H15 and H58 of the 23S rRNA between the bacterial and archaeal crystal structures, with no significant differences in their primary or secondary structure. Both helices are held in different orientations by nearby ds-proteins. In bacteria, helix H15 interacts with ds-proteins L9 and L28, while in
archaea it contacts ds-proteins L7Ae and L15e. Similarly, helix H58 has no nearby ds-proteins in bacteria, but in archaea it makes extensive contacts with ds-protein L37Ae. There are changes in the overall ribosomal interaction network as a result of the rearrangement of these two helices. While it is possible that the differences in the tertiary conformation of these two helices are crystallization artifacts, the interactions with ds-proteins make it likely that these are physically (perhaps physiologically) relevant differences between these domains of life.

In addition to ds-proteins per se, signatures distinguishing the domains are also evidenced by short indels in some universal r-proteins [22]. We have looked for coevolution between these indels and rRNA structural signatures by analyzing the covariation between the r-proteins and the rRNA using mutual information. Our analysis identified a bacterial-specific insertion (~12 amino acids in length) in the N-terminal domain of universal r-protein S4 that covaries with helix h16 of the bacterial 16S rRNA. Helix h16 was previously identified as one of the strongest signatures in the 16S rRNA that distinguished the archaea from the bacteria [5]. The structural signatures in S4 and helix h16 make exclusive contact with each other (Figure 2.7) and their coevolution is evidence that even within the universal r-proteins, characteristic signatures were evolving after the lineages diverged.

Ribosomal protein S4 is located near the decoding site of the ribosome and is a primary binding protein in the 30S subunit assembly map [23; 24]. Given this, one asks how the structural signatures in S4 and helix h16 of the 16S rRNA affect the translation process in the bacteria? One possibility is that, with S4’s position near the decoding site, the two signatures affect the dynamics of the region through increased interactions between S4 and helix h16. In fact, mutations in S4 are known to affect the precision of translation [25–27], though none occurring within its structural signature have, to our knowledge, been studied. Another

![Figure 2.7: Illustration of ribosomal protein S4 bound to the 16S rRNA in T. thermophilus. (a) A diagram of the secondary structure of the rRNA and its interactions with the protein in the vicinity of the binding region and (b) a 3D view of the structure in the same region. Colors are consistent between two as follows: the rRNA structural signature (green), the protein structural signature (orange), and the conserved core of S4 (yellow). Red spheres show the location of mRNA bound to the ribosome.](image)
possibility, suggested by preliminary studies of the folding of S4 in the presence of 16S rRNA, is that the 
two structural signatures may make the initial contacts in the docking of S4 to the 16S rRNA during the 
assembly of the bacterial SSU (Luthey-Schulten, unpublished data).

2.3.2 Evolution of ribosomal proteins L36 and L40e

Another general pattern to emerge from studying the ribosomal signatures is the relationship between do-
main specific r-proteins and the conserved core of the rRNA. There are a number of ds-proteins in the 
bacterial LSU that do not make contact with bacterial 23S rRNA structural signatures, but interact only with 
the structurally invariant rRNA. In the archaeal LSU, these ds-proteins are consistently replaced by either 
archaeal rRNA structural signatures or archaeal specific r-proteins (spatial analogs or r-protein extensions). 
The interaction network within the ribosome remains conserved between the two domains, even though the 
interactions are provided by different mechanisms. This observation leads us to predict an archaeal spatial 
analog to the bacterial specific r-protein L36.

Protein L36 binds to helices from domains II, V, and VI in the bacterial 23S rRNA (these are H42, H89, 
H91, and H97; see Figure 2.8(b)) and, in some of the E. coli LSU crystal structures, it makes contacts with 
the L11-arm (H43 and H44), to which r-proteins L10 and L11 bind. Two bacterial 23S rRNA sequence 
signatures (base pairs G2526:U2537 and A1032:G1122) make base specific contacts with r-protein L36, 
helping to establish its binding site in the bacteria. The analogous binding site is empty in the structure of 
the H. marismortui LSU, even though the structure of the rRNA in the region is highly conserved. L36 is 
known to be important for the structural stability of the bacterial LSU; deletion studies in E. coli have shown 
that its absence increases the accessibility of the region to reagents (dimethyl sulfate and hydroxyl radicals) 
and slows cell growth by 40–50% [28]. The lack of any archaeal specific 23S rRNA structural signatures 
near the binding site strongly suggests that the structural stabilization afforded to the bacterial LSU by L36 
must be, in this case, provided by archaeal specific r-protein interactions. Since there are no nearby archaeal 
ds-proteins in the H. marismortui LSU structure, the possibility of a yet unresolved archaeal spatial analog 
to L36 has to be considered.

The crystal structure of the H. marismortui LSU is missing only two of the fourteen ds-proteins whose 
genes are present in the species’ genome: L40e and LX. These are the only candidates for an archaeal spatial 
analog of L36, and of these two, only L40e is present in all archaeal sub-branches. While L40e is slightly 
longer than L36 (48 versus 38 amino acids on average), both proteins are highly basic and contain zinc 
finger motifs. A sequence comparison shows that the two proteins are not homologous. A solution structure 
of L40e has recently been determined by NMR spectroscopy [29], and the structures of L40e and L36 have 
a similar topology. From a superposition of L40e on L36 in its binding site in the bacterial LSU shown 
in Figure 2.8(c), it is clear that L40e fits into the cavity created by the junction of the four rRNA helices. 
Additionally, molecular dynamics simulations show that L40e is stable in this position in the archaeal LSU 
and provides interactions that could help to interconnect the 23S rRNA structure (data not shown).

Additional support for L36 having a spatial analog in the archaeal LSU comes from signatures in the 
genomic organization of the r-proteins. As discussed previously, only the three genes of bacterial specific 
r-proteins L17, L33, and L36 are located in the conserved clusters of universal r-protein genes. Like L36,
both L17 and L33 bind to conserved regions of the 23S rRNA with no nearby rRNA structural signatures. Both of these ds-proteins have known spatial analogs in the archaeal LSU (L31e and L44e, respectively). Assuming the shared organization of the genes of these three r-proteins correlates to other shared features,
we would again anticipate r-protein L36 to have a spatial analog.

While no single piece of the above evidence is by itself decisive, the consistency of the accumulated data within the signature framework implies that archaeal ds-protein L40e is the unresolved spatial analog to L36 in the archaeal LSU. Since the L11-arm appears to be open in the *H. marismortui* crystal structure, L40e may have been lost during the crystallization process. The presence of a ribosomal protein in this region of the archaeal LSU would have an impact on the dynamics of the ribosome during translation.

### 2.3.3 Final remarks

The emergence of the primary organismal lineages was a profound event in the evolution of life. Through our analysis of ribosomal signatures we have provided a glimpse into the evolutionary past, at the “base” of the UPT. This study has identified the ribosomal signatures and provided examples of how they are helpful in understanding the evolutionary dynamic by which the ribosome arose. These signatures give each phylogenetic domain a distinctive character and bespeak stages through which the evolution of the ribosome must have proceeded, both prior to the emergence of the individual lineages themselves (in the universal ancestral state), and subsequently, separately within each primary lineage.

### 2.4 Methods

All sequence and structural analyses, including identification of sequence and structural signatures, were performed using MultiSeq [30] and VMD [31].

#### 2.4.1 Creation of evolutionary profiles

The initial sequence alignments for the 16S and 23S rRNA were obtained from the Comparative RNA Web Site [6] and then curated to remove incomplete sequences. Environmental 16S rRNA sequence alignments were obtained from the Greengenes database [17]. Genomic data was obtained from the Integrated Microbial Genomes system [32]. Due to the length of the sequences and the differing levels of conservation in regions of the molecule, the 16S rRNA alignment was broken up into four sections by domain of the molecule, which correspond to nucleotides (*E. coli* numbering): 1-556, 557-912, 913-1396, and 1397-1534. The 23S rRNA was broken up similarly into the six domains defined by Ban *et al.* [33] (*E. coli* numbering): 1-531, 532-1268, 1269-1646, 1647-2018, 2019-2627, 2628-2902. Non-redundant sets of sequences that best represent their phylogenetic diversity, so-called evolutionary profiles, were then generated for each domain as described in Sethi *et al.* [34] and Roberts *et al.* [35]. The sizes of the profiles for the 16S rRNA were (97% percent identity cutoff/maximum-entropy cutoff): 273/14, 239/20, 226/24, and 70/27. The sizes of the profiles for the 23S rRNA were (97%-NR/ME-EP): 152/28, 135/19, 116/21, 107/17, 114/14, and 130/32. The evolutionary profiles were used in the identification of sequence and structural signatures, and for the mutual information analysis.
2.4.2 Identification of sequence and structural signatures

Sequence signatures between the bacteria and the archaea were identified as positions in the profile where the nucleotide identity was conserved in at least 90% of the sequences in one domain of life while being present in less than 10% of the sequences in the other domain. Positions that had a gap in more than 10% of the sequences in either domain were excluded.

To identify structural signatures, the 3D structures of 23S rRNAs from four organisms (H. marismortui (PDB code 1S72), D. radiodurans (PDB code 2D3O), E. coli (PDB code 2AW4), and T. thermophilus (PDB code 2J01)) were structurally aligned using STAMP [36] and regions of structural variation identified. These regions were then compared to a sequence profile and regions greater in length than one base that were characteristically different between at least 90% of the bacteria and the archaea were considered structural signatures. Additional hand checking was then performed to verify the structural signatures.

2.4.3 Sequence phylogenetic analysis of 16S and 23S rRNAs

The sequence based phylogenetic trees of the 16S and 23S rRNAs were based on full sequence alignments from the Comparative RNA Web Site [6].

The phylogenetic trees used for calculating the contribution of the sequence signatures to the separation between the bacterial and archaeal sub-branches in the 16S and 23S rRNA phylogenies were generated using RAxML-VI-HPC version 2.2.3 [37] under the GTRMIX model. For each analysis, 200 independent maximum likelihood searches were performed using a unique maximum-parsimony starting tree. The tree with the highest likelihood value was then evaluated for support using 1000 non-parametric bootstrap runs. All other options were used at their default value.

The sequence phylogenetic tree of the 23S rRNA shown in Figure 2.1 was generated using MrBayes version 3.1.2 [38]. The F81 model with gamma-distributed rate variation across sites for the nucleic acid. Starting with default settings, 4 simultaneous runs were performed containing 6 chains each. 1,000,000 trees were generated for each run and every 100th tree was sampled for analysis. The burn-in fraction was set to 0.5 and the number-of-swaps parameter to 4. The first 75% of each dataset was discarded before the analysis process. The tree was drawn using the program Unrooted (http://pbil.univ-lyon1.fr/software/unrooted.html).

2.4.4 Structural phylogenetic analysis of 23S rRNA

The structural phylogenetic trees of the 23S rRNA were based on structures of the molecule from four organisms: H. marismortui (1S72), D. radiodurans (2D3O), E. coli (2AW4), and T. thermophilus (2J01). Before the analysis began, regions which were unresolved in any of the structures were removed from all of the structures to eliminate any artificial indels that could result in artifacts in the resulting phylogenetic tree. These regions were (E. coli numbering): 1-14, 270-282, 652-656, 878-907, 1055-1108, 1534-1543, 1910-1924, 2106-2225, 2304-2311, and 2891-2904. The sequence alignment was used to define corresponding nucleotides so that the structural phylogenetic trees could be compared directly to sequence phylogenetic trees, which used the same alignment. The sequence alignment was also checked against a structure based alignment and found to be in excellent in agreement in the well-aligned regions, once again confirming the
power of comparative sequence analysis. Further investigation showed that the alignment between archaea and bacteria in regions near structural signatures could be improved using the structural data. The structural signatures in which definitive alignment improvements are possible are indicated in Table 2.2.

The structural phylogenetic trees were calculated using a measure of structural homology, $Q_H$ [10; 11], developed to test the congruence of phylogeny of structures in proteins. $Q_H$ was modified for this study to compare nucleic acid structures using the coordinates of the phosphorus atoms in the calculations in place of the $\alpha$-carbon coordinates typically used in protein calculations. $Q_H$ includes a score for the aligned regions and a gap penalty for the indels that is dependent on their spatial extent and length.

### 2.4.5 Mutual information analysis of S4 and 16S rRNA coevolution

Mutual information in conjunction with a contact filter was used to detect coevolution of r-protein S4 and the 16S rRNA. First, an evolutionary profile was created for S4 as described in [34]. The mutual information of each position in the S4 profile with each position in the RNA profile was then calculated using the formula: $MI(P_i, N_j) = H(P_i) + H(N_j) - H(P_i, N_j)$, where $MI(P_i, N_j)$ is the mutual information of amino acid $i$ in the protein and base $j$ in the nucleic acid. $H(X_i)$ is Shannon’s entropy, given by $H(X_i) = -\sum_{t=1}^{n} p(x_t^i) \log_2 p(x_t^i)$ with $p(x_t^i)$ being the probability of position $i$ in the alignment being of type $t$ (n=21 for proteins and n=5 for nucleic acids, including a gap as a distinct character). The joint entropy is given by, $H(X_i, Y_j) = -\sum_{t_1=1}^{n} \sum_{t_2=1}^{n} p(x_t^{i_1}, y_j^{i_2}) \log_2 p(x_t^{i_1}, y_j^{i_2})$, where $p(x_t^{i_1}, y_j^{i_2})$ is the probability of position $i$ in the first alignment being of type $t_1$ and position $j$ in the second alignment being of type $t_2$. Amino acids or bases that did not occur in a column of the alignment were give zero contribution to the entropies (i.e., $0 \cdot \log_2 0$ was taken to be 0). Columns with greater than 50% probability for gaps were neglected. The mutual information values where then normalized by the joint entropy [39] and those amino acids in S4 that had mutual information with a nucleotide in the 16S rRNA more than four standard deviations above average and that were within 15 Å of each other in the E. coli structure were singled out for manual investigation.

### 2.4.6 Molecular dynamics simulation of r-protein L40e in the archaeal LSU

A molecular dynamics simulation of a model of r-protein L40e bound to the archaeal LSU was performed to assess the stability of the complex. The atomic coordinates of the 23S rRNA were taken from the crystal structure of the archaeal LSU (H. marismortui; PDB code 1S72) and the coordinates of L40e from an NMR solution structure [29] (Sulfolobus solfataricus; PDB code 2AYJ). Only the well-structured residues of L40e were used (18-48); the unstructured N- and C-terminal regions were removed. The position of L40e in relation to the 23S rRNA was determined by structural similarity with r-protein L36 in the bacterial LSU (see Results and Discussion in the main text). Only the following regions of the 23S rRNA near the binding site were included in the model (H. marismortui numbering): 1115-1240, 2480-2590, 2675-2710, and 2766-2810. Each segment of rRNA was terminated using the standard 5TER and 3TER termination states. The four zinc-coordinating cysteine residues of the C$_4$ zinc finger motif in L40e were deprotonated based on their distances in the NMR structure of the protein [40]. Magnesium ions that made contact with r-protein L36 in the T. thermophilus LSU (PDB code 2J01) were included in the model. The system was solvated
and ionized according to Eargle et al. [41] and hydrogen atoms were added using the PSFGEN plug-in of VMD [31].

During the simulation, all atoms of the rRNA (except those in the L11-arm – nucleotides 1132-1230 and 2772-2802) were harmonically constrained to their positions with a force constant of 5 pN/Å². Multiple steps of conjugate gradient minimization were performed followed by 2 ns of simulation using the molecular dynamics program NAMD2 [42] and the CHARMM force field [43; 44]. The simulation was carried out in the NPT ensemble at a temperature of 300K and a pressure of 1 bar with periodic boundary conditions. An integration time step of 1 fs was used. Non-bonded interactions were calculated with a switching distance of 10 Å, a cutoff of 12 Å, and PME for long-range electrostatics.
Chapter 3

Communal Sharing of Ribosomal Proteins among the Early Bacteria

3.1 Background

The ribosome is an elaborate ribonucleoprotein complex whose evolution is intrinsically linked with that of the cell. It has been recognized since the 1970’s that the molecular core of the ribosome was in place before the divergence of the three primary organismal lineages, Bacteria, Archaea, and Eucarya (the domains of life). The history of these lineages, as inferred from the rRNA and represented by the UPT [45], provides an organismal reference by which the evolutionary history of a gene can be studied. Despite conservation of a large portion of the ribosomal structure among the lineages, the ribosomes of each domain of life contain certain sequence and structural signatures that are unique to and constant within the domain. Such signatures have been identified in both the rRNA and r-proteins, including many r-proteins that are specific to one of the primary lineages. This suggests that both large and small scale changes in the ribosome were still evolving after the domains diverged and then spreading among all of a domain’s developing sub-branches [46]. The exact mechanism by which homogenization of the branches might have occurred is unclear and certainly a matter of some debate, but pervasive HGT among aboriginal cellular life [47; 48] is one possible mechanism. Studying the pattern by which such a signature spread among the evolving lineages can help resolve the dynamics of the evolutionary process at the time.

Horizontal gene transfer, the acquisition of non-inherited genetic material, is widely regarded as a common and important evolutionary phenomenon [49–54]. It is now understood that HGT allows microorganisms to break out of strictly clonal, bifurcating lineages in their search for genetic innovation [55]. Despite the complexity of the ribosome and the potential for malfunction from acquiring a new version of a single ribosomal component, r-protein genes are known to have been horizontally transferred within a domain of life, although no inter-domain HGT has been identified. The first instance of HGT of an r-protein gene was reported by Brochier et al. [56] for S14. In their study, they classified the bacterial S14 sequences into distinct groups based on characteristic indels and presented phylogenetic evidence that, in some cases, the groups were at odds with the classical bacterial phylogeny. They argued that these discrepancies, as well

---

The contents of this chapter are based in part on work previously published as Ke Chen†, Elijah Roberts†, and Zaida Luthey-Schulten. “Horizontal gene transfer of zinc and non-zinc forms of bacterial ribosomal protein S4,” BMC Evol. Biol., 9:179 (2009). † denotes equal contributions.
as unusual gene ordering and duplications in the affected lineages, were the result of ancient HGT events. Thus, they proposed that there must have been some evolutionary pressure favoring the fixation of the transferred r-protein gene, in accordance with the “complexity hypothesis” of Jain et al. [57] regarding the lower probability of HGT for informational genes.

A later bioinformatics study further extended analysis of HGT and gene duplication in the r-proteins. Using genomes of thirty bacteria and genomic data for r-proteins of mitochondria and chloroplasts from seven eukaryotic organisms, Makarova et al. [58] found six additional ribosomal proteins (S18, L28, L31, L32, L33, L36) that shared similar evolutionary patterns to S14 within the bacterial lineage, including discrepancies in genome organization and gene copy number. Furthermore, they showed that the phylogenetic patterns were related to the zinc binding ability of the r-proteins. Two variants were found of each r-protein: one containing a zinc finger motif with four conserved cysteine residues (or occasionally three cysteine and one histidine residue) and another with a complete or partial disruption of the motif. The two variants were referred to as C+ and C-, respectively. Their data suggested that in each case the C+ variant was the ancestral form and that ancient gene duplication followed by disruption of the zinc finger in the paralog and later loss of the original C+ gene in some lineages (differential gene loss; DGL) was the major evolutionary pattern with HGT also occasionally occurring.

Following initial identification of C- variants of bacterial zinc-binding r-proteins, other laboratories began investigating their regulation in organisms with both C+ and C- genes to better understand the evolutionary pressures giving rise to the C- forms. It was predicted theoretically [59] and then found experimentally, first in *Bacillus subtilis* for S14 and L31 [60; 61] and then in *Streptomyces coelicolor* for S14, L28, L31, L32, L33, and L36 [62] and in *Mycobacterium tuberculosis* for S14, S18, L28, and L33 [63], that the paralogous C- versions of some r-protein genes were up-regulated under conditions of low zinc. These groups proposed that the C- paralogs served two possible functions, release of free zinc into the cell in low zinc environments (by ribosomal exchange with the endogenous C+ protein) and/or continued production of ribosomes under zinc-limiting conditions. Whether these are the only pressures that gave rise to the C- forms is unknown, but it is clear that some ribosomal proteins have a unique and interesting evolutionary history related to zinc binding.

All seven zinc binding r-proteins discussed above, except for S14, are unique to *Bacteria*. Such domain specific r-proteins are signatures of the bacterial ribosome. Roberts et al. [46] showed that the signatures are not limited to just entire domain specific ribosomal proteins, but can also take the form of domain specific insertions in the r-proteins that are universally distributed among all three domains of life. Such a case is found in the universal r-protein S4, a two domain protein ~200 amino acids in length that is essential for the initiation of SSU ribosomal assembly and translational accuracy. The C-terminal domain of S4 (residues 46–206; all residues given in terms of *E. coli* numbering) is known to be an RNA binding domain, binding to both rRNA and messenger RNA (mRNA) [64–66], and is homologous between *Bacteria* and *Archaea*. While the overall sequence identity for the C-terminal domain is only 36% among bacteria and 32% across all domains of life, the region making contact with the ribosomal RNA is conserved with an average sequence identity of 46% and 40% respectively. The N-terminal domain, in contrast, appears to be non-homologous between *Bacteria* and *Archaea* and was identified in [46] as a bacterial-specific
structural signature that coevolved with a bacterial specific extension of an RNA helix (helix h16) on the 16S rRNA. Alone, the bacterial S4 structure has been determined only without the unstructured N-terminus [67], but when complexed with the ribosome the structure of the full protein has been determined. The crystal structure of the T. thermophilus ribosome [68] shows that the N-terminal domain of S4 contains a zinc finger motif ligated to a zinc atom and the sequence analysis presented here shows conservation of the four cysteine residues in the zinc-finger motif only in a subset of the bacterial lineages. This variation in zinc binding ability within the bacterial lineages of S4 was overlooked in previous studies of the evolutionary history of zinc-binding r-proteins.

The recent growth in the number of available bacterial genome sequences allows a broad evolutionary history of a gene to be reconstructed, especially in regard to HGT [69; 70]. Besides sequence data for phylogenetic reconstructions, full genomes provide data on genome organization and gene distribution, which are particularly useful in aiding interpretation of possible HGT events. In this study, we use 660 available bacterial genomes to study the evolution of ribosomal protein S4 in the bacteria. We find that S4 can be classified into C+ and C- variants (zinc binding and non-zinc binding, respectively), with multiple independent origins of the C- form. A maximum likelihood tree of S4 shows disagreement with the standard bacterial phylogeny, indicating a more complex evolutionary history than previously known. Considering the fact that the S4 gene is part of a highly conserved gene cluster in bacteria consisting of the S10–spc–α operons [71], we see surprising evidence for the endogenous origin of the C- form in some phyla and hypothesize that both the C+ and C- forms may have been present before the bacterial phyla diverged with different lineages sampling from the variants according to the local environment. In accordance with this hypothesis, we also present evidence that C- paralogous copies in genomes containing both variants of S4, as well as all S4 genes outside the α-operon, are results of HGT events. Regulation of the paralogous S4 genes seems to differ from the zinc-binding r-proteins previously identified, and the expanded distribution of the C+/C- variants in all the zinc-binding r-proteins we present may provide insight on the evolution of zinc usage in bacterial lineages.

3.2 Classification and Phylogeny of Ribosomal Protein S4

3.2.1 Sequence alignment and classification of bacterial r-protein S4

To study the history of S4 in bacteria, we first extracted 688 sequences of S4 and paralogs from 660 complete and draft bacterial genomes. We then constructed a multiple sequence alignment (MSA) of the sequences using automated alignment tools followed by manual correction (see Methods). Any evolutionary study of a large set of diverse organisms is likely to reveal a complex history, so to aid further analysis of the relationships between the sequences we classified them into six types using key sequence and structural signatures that define apparently monophyletic groups (such features are also known as synapomorphies). These types classify the sequences according to the presence of or disruption pattern in the zinc finger motif. We find one C+ version (with four conserved cysteines) and five C- subtypes (with various patterns of loss of the zinc finger). C-(I), C-(II) and C-(III) sequences possess a seven residue indel present in the C+ type but show gradual loss of the four cysteines (from two to one to zero). C-(IV) and C-(V) subtypes are missing the
Figure 3.1: Multiple sequence alignment of ribosomal protein S4. Shown is a representative sample of the full sequence alignment. Sequences are grouped according to specific sequence characteristics (see text) and positions are colored by conservation within the group at (blue) 95% and (red) 70%. The arrows above the alignment indicate positions of the two pairs of cysteine residues. The three-letter abbreviations indicate the phylum or class that the organisms belong to: AC (Acidobacteria), ACT (Actinobacteria), ALP (Alphaproteobacteria), AQF (Aquificae), BAT (Bacteroidetes), BET (Betaproteobacteria), CHF (Chloroflexi), CHL (Chlamydiae), CHR (Chlorobi), CLT (Clostridia), CYN (Cyanobacteria), DEL (Deltaproteobacteria), DTH (Deinococcus-Thermus), EPS (Epsilonproteobacteria), FUS (Fusobacteria), GAM (Gammaproteobacteria), MAG (Magnetococcus), MOL (Mollicutes), PLN (Planctomycetes), SPR (Spirochaetes), VER (Verrucomicrobia). (Figure courtesy of K Chen)

indel characteristic of the C+ type as well as all four cysteines. Further distinctions between the C- subtypes are based on sequence signatures in the N-terminus. Figure 3.1 shows a sample of the N-terminal portion of the alignment from all of the major bacterial phyla grouped according to these classifications.

From a conservation analysis of the MSA, it is apparent that the S4 sequences can be broadly classified into C+ and C- variants (following the notation introduced by Makarova et al. [58]) based on the conservation of four cysteine residues in the N-terminal domain. C+ type sequences contain two conserved pairs of cysteine residues in a “CXXC...CXXXCC” motif. The first pair appears near the beginning of the sequence (at residues 9 and 12) and the second pair in a seven residue segment that is an insertion relative to most of the C- sequences (the first red block in Figure 3.1). As shown in the T. thermophilus ribosome structure, these four cysteine residues bind a zinc ion. Within the C+ group, the N-terminal domain is highly conserved
with an average percent sequence identity of 65%. The C+ group includes sequences from diverse bacteria groups: Acidobacteria, Actinobacteria, Chloroflexi, Clostridia, Cyanobacteria, Deinococcus-Thermus, Planctomycetes, Proteobacteria (Beta and Delta classes) and Thermotogae.

The C- variants of S4 show less homogeneity than their C+ counterparts. By definition, they all lack the four cysteine residues, but other characteristic features in the N-terminus allow them to be further classified according to their likely evolutionary origin. The major distinguishing feature between the C- forms is the presence or absence of the seven residue indel that contains the second pair of cysteine residues in the C+ form. Three C- types, C-(I), C-(II), and C-(III), possess the indel, but have disruption of the zinc binding motif. Each of these types is confined to a small portion of the bacterial tree, while sequences containing the indel with the conserved cysteine residues are seen in a wide variety of bacteria. This difference suggests that each of these three groups may have been formed by relatively recent, independent mutations of an ancestral C+ form.

To test this hypothesis, we performed a phylogenetic analysis of the sequences in the C-(I), C-(II), and C-(III) groups relative to the C+ sequences. The first group, C-(I), includes the S4 sequences from most of Betaproteobacteria and all of Gammaproteobacteria. The remaining Betaproteobacteria are all of the C+ type, and, interestingly, the C-(I) Betaproteobacteria show a gradual loss of the four cysteine residues from two to one and, finally, to zero. The Gammaproteobacteria also show a distribution of two, one, or zero cysteine residues. The C-(I) sequences have also lost a three residue turn (the second red block in Figure 3.1) compared to all of the other variants of S4. Figure 3.2 shows a maximum-likelihood (ML) reconstruction of the phylogenetic history of S4 in Proteobacteria except Alphaproteobacteria, which lack the seven residue indel. In the tree, Beta- and Gammaproteobacteria share a common ancestor containing the C+ variant of S4 and the root of the Proteobacteria also appears to have been a C+ type S4. The most parsimonious explanation for the origin of the C-(I) form appears to be that a single evolutionary event, characterized by the deletion of the three residue turn and loss of the zinc-binding motif, occurred in the Betaproteobacteria lineage and was inherited monophyletically by the descendant Betaproteobacteria and the Gammaproteobacteria.

The C-(II) group contains all and exclusively sequences from the Epsilonproteobacteria. These sequences do contain the three residue turn that the C-(I) Beta- and Gammaproteobacteria sequences are missing. Additionally, the pattern of disruption in the zinc-binding motif is markedly different from the C-(I) group. In C-(II) sequences, the four cysteine residues are consistently replaced by two glutamic acid residues, one arginine residue, and one serine residue, as opposed to the variety of residues seen in C-(I). Differences in both the pattern of indels and of motif disruption suggest an independent origin for the C-(II) form and phylogenetic analysis supports this interpretation. In the tree shown in Figure 3.2, Epsilonproteobacteria branches outside of the C-(I) group, appearing to diverge near the root of Proteobacteria. The low bootstrap values at higher branch points do cast uncertainty as to whether Epsilonproteobacteria diverged from a common Proteobacteria ancestor or directly from the bacterial root. In either case, however, the C-(II) sequences would be a result of an independent mutation event in an ancestral C+ form that occurred after the divergence of the Epsilonproteobacteria lineage. Conservation of the “EXXE...RXXXXXS” motif suggests that a salt bridge may have replaced the zinc finger as a structural element in the C-(II) S4
Figure 3.2: Consensus phylogenetic tree of ribosomal protein S4 in Proteobacteria. The phylogenetic tree for Proteobacteria (except Alphaproteobacteria) was constructed from 1000 maximum-likelihood inferences and rooted using Deinococcus-Thermus and Thermotogae as outgroups. Branches that are monophyletic with respect to a class or order are collapsed with the number of taxa in the branch given in parentheses. Node label are bootstrap proportions estimated from 5000 replicates. The scale bar represents one change per site.

sequences, and threading of an epsilonproteobacterial sequence onto the T. thermophilus crystal structure of S4 confirms that the residues would be properly oriented.

The final C- type containing the seven residue indel, C-(III), is made up of S4 sequences from a subset of Spirochetes: the genus Leptospira. All other Spirochetes currently sequenced lack the indel in r-protein S4. C-(III) sequences have a zinc disruption pattern of “VXXM...LXXXXS” or “VXXM...FXXXXF” and do have the three residue turn missing in C-(I). Additionally, there are numerous sequences signatures separating the C-(III) sequences from those in either group C-(I) or C-(II). Phylogenetically, these sequences appear to branch directly from the root of the C+ form, no further relationships can be resolved. Since the C-(III) group appears to monophyletically descend from an ancestral C+ form, we consider that it too was an independent evolution of zinc disruption in S4.
All of the remaining S4 C- sequences lack the seven residue indel and both pairs of cysteine residues. C-(IV), the largest C- group, consists of r-protein S4 sequences from a wide variety of bacteria: *Actinobacteria, Alphaproteobacteria, Aquificae, Bacilli, Bacteroidetes, Betaproteobacteria, Chlamydiae, Chlorobia, Chloroflexi, Cyanobacteria, Deinococcus-Thermus, Deltaproteobacteria, Fusobacteria, Mollicutes, Planctomycetes, Spirochaetes*, and *Verrucomicrobia*. The N-terminal domains of the C-(IV) sequence are much less conserved than the C+ form, having an average percent identity of 36%, and do not contain any characteristic sequence or structural signatures by which they could be further classified.

A small number of *Clostridia* sequences (17) constitutes the last defined type, C-(V). These C- sequences lack the seven residue indel, but are different from C-(IV) sequences (and all other S4 sequences) in that they are missing a “PGXHG” motif starting at residue 38. This motif is highly conserved in the other S4 sequences and is unambiguously alignable across all other groups. In C-(V) *Clostridia* sequences, this region is 2–4 residues shorter and cannot be reliably aligned to the other types. All but one of the S4 sequences in this group are from genomes that also contain a C+ type S4.

### 3.2.2 Phylogenetic reconstruction of S4’s evolutionary history

The evolutionary history of S4 was analyzed using a ML phylogenetic reconstruction of all the sequences of r-protein S4 and its paralogs obtained from the 660 genomes, as described in Methods. Figure 3.3 shows an unrooted phylogenetic tree obtained from a consensus of 1000 ML trees. Like many phylogenetic reconstructions using a large number of sequences, branch points above the bacteria phyla level are difficult or impossible to reliably determine [72] and most branches appear to radiate from a few ancestral points in our consensus tree. Trees of the C-terminal RNA binding domain and the N-terminal bacterial specific domain were also generated separately using the same method (data not shown). The C-terminal tree had similar branchings as the tree shown in Figure 3.3, but with fewer well-supported branches near the bacterial root. The N-terminal domain, however, is too short to draw any reliable conclusions regarding its relative contribution to the phylogenetic signal.

The consensus phylogenetic tree of the entire protein shows good agreement with the classifications of S4 that we introduced earlier. It is roughly divided into two central foci, one representing the C+ form (white) and the other the C-(IV) form (blue), although a few C-(IV) lineages branch within the C+ half. C-(I), C-(II), and C-(III) (gray) are recent, independent mutations of an original C+ form (discussed above), and we treat them as C+ for the remainder of the discussion. The C-(V) form (yellow) is a monophyletic branch descending from the C-(IV) root.

Within the C+ branch of the tree, three bacteria phyla are monophyletic with high support values and yet contain both C+ and C-(IV) forms: *Actinobacteria, Chloroflexi* and *Deinococcus-Thermus*. In each case, there are two branches descending from the phylum that are monophyletic, one with respect to C+ and one to C-(IV). Despite the sequences in each C+ branch containing the seven residue indel characteristic of the C+ group and sequences in each C-(IV) branch lacking the indel, the branches have a higher average sequence identity (56%, 46%, 60%, respectively) than in general would be expected for a C+ and a C-(IV) group (~40%). We therefore consider it likely that these are real phylogenetic branches and not artifacts, particularly for *Actinobacteria* and *Deinococcus-Thermus*.  

25
Figure 3.3: Consensus unrooted phylogenetic tree of S4. The tree was constructed from 1000 maximum-likelihood inferences. Node labels are bootstrap proportions estimated from 5000 replicates. Branches that are monophyletic with respect to a phylum or class and also with respect to a sequence classification have been collapsed, parentheses give the number of sequences in the branch. Colors indicate the sequence classifications: (white) C+, (gray) C-(I) – C-(III), (blue) C-(IV), and (yellow) C-(V). Sequences from genomes with multiple divergent copies of an S4 gene are marked with a (†) dagger. The two sequences marked with an (*) asterisk are identical copies resulting from large-scale genome duplication.

Also within the C+ branch, the C-(IV) Chlorobi and C+ Epsilonproteobacteria group together, with an average percent identity of 50%. However, the Chlorobi sequences have nearly as high average percent identity with C-(IV) groups (46%) and the support value of 52 is fairly low, so this grouping may be a reconstruction artifact.

Comparing the S4 phylogenetic tree with the classical bacterial phylogeny, many bacterial groups show good agreement with the tree at the phyla level: The phyla Aquificae, Bacteroidetes, Chlamydiae, Tenericutes (Mollicutes) and Verrucomicrobia in the C-(IV) branch and Acidobacteria, Chlorobi, Chloroflexi, Deinococcus-Thermus, Fusobacteria, and Thermotogae in the C+ branch are all monophyletic with high
support values in the tree. A few other groups, most notably the **Proteobacteria** and the **Firmicutes**, are monophyletically supported at the class level. **Proteobacteria** classes Beta/Gamma-, Delta-, and Epsilon-proteobacteria independently meet at the root of the C+ branch, while Alphaproteobacteria originates in the C-(IV) branch. In the **Firmicutes**, Clostridia (non-paralogs) branches from C+ while Bacilli branches from C-(IV). The remaining bacteria phyla, Actinobacteria, Cyanobacteria, Planctomycetes, and Spirochaetes, have more convoluted branching patterns, with members branching in either the C+ or C-(IV) group with little regard for classical phylogeny. Disagreement with the classical bacterial phylogeny is an indication that a process more complex than standard vertical inheritance occurred with r-protein S4 during bacterial evolution.

### 3.3 Horizontal Gene Transfer of S4

#### 3.3.1 Identification of paralogous and duplicated S4 genes

A key element that led to the identification of HGT and gene duplication with DGL in previous bacterial r-proteins was the analysis of genomes containing multiple copies of the r-protein genes [58]. Among the 660 bacterial genomes in our study, 26 organisms from the groups Clostridia, Betaproteobacteria, Deltaproteobacteria, Gammaproteobacteria, Actinobacteria, Spirochaetes and Planctomycetes possess more than one copy of the S4 gene (marked with a dagger symbol on the tree in Figure 3.3). Most have two copies, as shown in Table 3.1, and normally one copy is a C+ form and the other a C-(IV) or C-(V). Two Clostridia, A. metalliredigens and C. acetobutylicum, have three copies of the S4 gene, both have one C+ and two C-(V) variants. The genomes of Methylobacillus flagellatus, Psychromonas ingrahamii, and Leptospira borgpetersenii do not match the above pattern in that both genes are of the same type. However, in each of these three cases the sequence identities of the two copies are extremely high, 100%, 99%, and 100%, respectively. These cases are undoubtedly recent gene duplication events. In fact, P. ingrahamii and L. borgpetersenii have duplicated a large segment of their conserved operon cluster. M. flagellatus is known to have a large 140 kbp repeat in its genome [73], this repeated region contains the S4 gene.

Usually in a case of two divergent copies of a gene in a genome, one copy is the original and the other a paralog, either from an ancient gene duplication or from an HGT event. Without experimental evidence of activity, determining which is the active gene and which the paralog can often be problematic. In the case of r-protein S4, however, the genome content can provide evidence to make a determination: many ribosomal protein genes in bacteria are known to be located in conserved gene clusters. The gene for S4 is usually located in a cluster along with the genes for ribosomal proteins S13, S11, and L17 and the gene for the RNA polymerase alpha subunit, which together are known as the α-operon because they are co-regulated in *E. coli*. If two copies of the S4 gene are present in a genome with one copy inside the α-operon and the other outside it, we assume the copy inside the α-operon is the original form and the other the paralog. In every genome containing **two divergent** S4 genes, the C+ form is located in the α-operon and the C-(IV) or C-(V) form outside. Using the above criteria, we conclude that the C+ form is the original S4 sequence and the C- form the paralog in these genomes. It then remains to determine the origin of the paralogous C- sequences, whether by HGT or gene duplication.
Table 3.1: Genomes containing multiple copies of the S4 gene

<table>
<thead>
<tr>
<th>Organism Name</th>
<th>Taxon</th>
<th># copies</th>
<th>PID(%)</th>
<th>Classification</th>
<th>Other Zn-ribbon duplicates&lt;sup&gt;a&lt;/sup&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Frankia</em> sp. EAN1pec</td>
<td>Actinobacteria</td>
<td>2</td>
<td>40.57</td>
<td>C+&lt;sup&gt;b&lt;/sup&gt;; C-(IV)</td>
<td></td>
</tr>
<tr>
<td><em>Salinispora arenicola</em> CNS-205</td>
<td></td>
<td>2</td>
<td>36.79</td>
<td>C+; C-(IV)</td>
<td>S14, L33, L28&lt;sup&gt;c&lt;/sup&gt;, L31, L32</td>
</tr>
<tr>
<td><em>Salinispora tropica</em> CNB-440</td>
<td></td>
<td>2</td>
<td>36.32</td>
<td>C+; C-(IV)</td>
<td>S14, L33, L28&lt;sup&gt;c&lt;/sup&gt;, L31, L32</td>
</tr>
<tr>
<td><em>Methyllobacillus flagellatus</em> KT</td>
<td>β-proteobacteria</td>
<td>2</td>
<td>100.0</td>
<td>both C-(IV)</td>
<td>L36</td>
</tr>
<tr>
<td><em>Nitrosomonas europaea</em> ATCC 19718</td>
<td></td>
<td>2</td>
<td>36.62</td>
<td>C+; C-(IV)</td>
<td></td>
</tr>
<tr>
<td><em>Psychromonas ingrahamii</em> 37</td>
<td>γ-proteobacteria</td>
<td>2</td>
<td>99.51</td>
<td>both C-(I)</td>
<td>L36&lt;sup&gt;d&lt;/sup&gt;</td>
</tr>
<tr>
<td><em>Bdellovibrio bacteriovorus</em>  HD100</td>
<td>δ-proteobacteria</td>
<td>2</td>
<td>39.35</td>
<td>C+; C-(IV)</td>
<td></td>
</tr>
<tr>
<td><em>Myxococcus xanthus</em> DK 1622</td>
<td></td>
<td>2</td>
<td>36.41</td>
<td>C+; C-(IV)</td>
<td>S14, L33, L28&lt;sup&gt;c&lt;/sup&gt;</td>
</tr>
<tr>
<td><em>Sorangium cellulosum</em> ‘So ce 56’</td>
<td></td>
<td>2</td>
<td>42.45</td>
<td>C+; C-(IV)</td>
<td></td>
</tr>
<tr>
<td><em>Alkaliphilus metalliredigens</em> QYMF</td>
<td>Clostridia</td>
<td>3</td>
<td>38.53&lt;sup&gt;e&lt;/sup&gt;</td>
<td>C+; C-(V)&lt;sup&gt;f&lt;/sup&gt;</td>
<td></td>
</tr>
<tr>
<td><em>Alkaliphilus oremlandii</em> OhILAs</td>
<td></td>
<td>2</td>
<td>40.00</td>
<td>C+; C-(V)</td>
<td></td>
</tr>
<tr>
<td><em>Clostridium acetobutylicum</em> ATCC 824</td>
<td></td>
<td>3</td>
<td>38.61&lt;sup&gt;e&lt;/sup&gt;</td>
<td>C+; C-(V)&lt;sup&gt;f&lt;/sup&gt;</td>
<td></td>
</tr>
<tr>
<td><em>Clostridium botulinum</em> A str. ATCC 3502</td>
<td></td>
<td>2</td>
<td>38.79</td>
<td>C+; C-(V)</td>
<td></td>
</tr>
<tr>
<td><em>Clostridium botulinum</em> A3 str. Loch Maree</td>
<td></td>
<td>2</td>
<td>37.38</td>
<td>C+; C-(V)</td>
<td></td>
</tr>
<tr>
<td><em>Clostridium botulinum</em> A str. ATCC 19397</td>
<td></td>
<td>2</td>
<td>38.79</td>
<td>C+; C-(V)</td>
<td></td>
</tr>
<tr>
<td><em>Clostridium botulinum</em> A str. Hall</td>
<td></td>
<td>2</td>
<td>38.79</td>
<td>C+; C-(V)</td>
<td></td>
</tr>
<tr>
<td><em>Clostridium botulinum</em> B1 str. Okra</td>
<td></td>
<td>2</td>
<td>38.79</td>
<td>C+; C-(V)</td>
<td></td>
</tr>
<tr>
<td><em>Clostridium botulinum</em> F str. Langeland</td>
<td></td>
<td>2</td>
<td>39.25</td>
<td>C+; C-(V)</td>
<td></td>
</tr>
<tr>
<td><em>Clostridium kluyveri</em> DSM 555</td>
<td></td>
<td>2</td>
<td>39.25</td>
<td>C+; C-(V)</td>
<td></td>
</tr>
<tr>
<td><em>Clostridium novyi</em> NT</td>
<td></td>
<td>2</td>
<td>40.38</td>
<td>C+; C-(V)</td>
<td></td>
</tr>
<tr>
<td><em>Clostridium perfringens</em> str. 13</td>
<td></td>
<td>2</td>
<td>38.79</td>
<td>C+; C-(V)</td>
<td></td>
</tr>
<tr>
<td><em>Clostridium perfringens</em> ATCC 13124</td>
<td></td>
<td>2</td>
<td>38.79</td>
<td>C+; C-(V)</td>
<td></td>
</tr>
<tr>
<td><em>Clostridium perfringens</em> SM101</td>
<td></td>
<td>2</td>
<td>38.79</td>
<td>C+; C-(V)</td>
<td></td>
</tr>
<tr>
<td><em>Leptospira borgpetersenii</em> serovar</td>
<td>Spirochaetes</td>
<td>2</td>
<td>100.0</td>
<td>both C-(III)</td>
<td>S14, L36&lt;sup&gt;g&lt;/sup&gt;</td>
</tr>
<tr>
<td><em>Hardjo-bovis</em> L550</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><em>Gemmata obscuriglobus</em> UQM 2246</td>
<td>Planctomycetes</td>
<td>2</td>
<td>40.95</td>
<td>C+; C-(IV)</td>
<td></td>
</tr>
</tbody>
</table>

<sup>a</sup>Observations based on annotation.
<sup>b</sup>Classification written in bold indicates that it is inside the α-operon.
<sup>c</sup>The duplicates of the zinc-binding r-proteins S14, L28, L33 sit together in the genome.
<sup>d</sup>Resulting from the duplication of the whole α-operon.
<sup>e</sup>Average value of the percent identities of the two pairs of in-operon and out-of-operon copies.
<sup>f</sup>*Alkaliphilus metalliredigens* and *Clostridium acetobutylicum* have 2 copies of the C-(V) form.
<sup>g</sup>Resulting from the duplication of the entire s10-spc and α-operon.
One clear-cut case of HGT appears to have occurred in the *Proteobacteria*. One *Beta*- and three *Deltaproteobacteria* have S4 paralogs that group within the C-(IV) branch of the phylogenetic tree in Figure 3.3 and C+ genes in the α-operon. Since the vast majority of *Beta*- and *Deltaproteobacteria* have only a single C+ copy of S4, we consider it unlikely that this pattern resulted from an ancient gene duplication that was lost in all *Beta*- and *Deltaproteobacteria* except these four organisms. Given the high support values near the branch with *Cyanobacteria*, we find it more likely that these organisms obtained the gene through a horizontal transfer from *Cyanobacteria*, although the S4 sequence is not similar enough to any available sequences for the specific source organism to be determined. *Betaproteobacteria* species *M. flagellatus*, which possesses only two copies of C-(IV) outside of the α-operon, also groups nearby in the tree and also likely received its C-(IV) S4 gene from *Cyanobacteria* before its large-scale genome duplication occurred. It must have lost its original C+ gene subsequent to the HGT, as it is no longer present in the α-operon.

Another example of probable HGT, albeit with a more complex pattern, is seen in the *Clostridia*. All *Clostridia* except one, *Finegoldia magna*, contain the gene for the C+ form of S4 in the α-operon. Fifteen *Clostridia* also contain a gene for the C-(V) form of the S4 sequence. Figure 3.4 shows an expansion of these two *Clostridia* branches from the consensus phylogenetic tree. In the C+ branch, all of the organisms with multiple copies of S4 are descended from a single branch. If we assume that the C+ form represents the
vertical phylogeny in this group, then the pattern is consistent with either a single HGT event in the ancestor of the Clostridium and Alkaliphilus genera with later differential loss in a few branches; or with three later HGT events, one for Alkaliphilus, a second for Clostridium perfringens, and a third for the branch containing the organisms Clostridium botulinum/kluyveri/acetobutylicum/novyi. However, two organisms possess three copies of the S4 gene, and the percent identities between the two C-(V) copies are 60.3% and 54.3% for A. metalliredigens and C. acetobutylicum, respectively. So there may have been an additional recent HGT of the transferred C-(V) gene. The support values are too low to allow a determination the source of this recent HGT, but additional genomes of related organisms could shed light on the history of the C-(V) form of S4.

In the two remaining lineages with genomes containing multiple S4 genes, multiple occurrences are relatively rare. Of the four Planctomycetes genomes available, one contains both the C+ gene in the α-operon and C-(IV) out of it while the other three contain only C-(IV) out of the operon. The low number of available Planctomycetes genomes sequenced makes it impossible to reconcile the origin of the paralog using a parsimony argument. In the Actinobacteria, the three genomes with two copies of the S4 gene can be accounted for by two recent HGT events (see Figure 3.5), one in the Salinispora genus and the other in the species Frankia sp. EAN1pec. Both of these paralogous genes appear to have originated in a Streptomyces source. The two Salinispora species also have acquired paralogs of five other zinc-binding r-proteins.
3.3.2 Regulation of C- S4 paralogs

To fully understand the evolutionary pressure giving rise to paralogous genes, it is helpful to know their regulation mechanism, especially when the two copies have similar functions. For the previously known zinc-binding ribosomal proteins, experiments on specific bacteria have shown that the paralogs can be turned on and off in response to zinc conditions. In *B. subtilis* (S14, L31) [61], *M. tuberculosis* (S14, S18, L28, L33) [63], and *S. coelicolor* (S14, L28, L31, L32, L33A, L33B, L36) [62], the C- paralogs were found to be expressed only under low zinc conditions. Their regulation was controlled by the zinc uptake regulator (Zur) transcription factor, except for L33B and L36 in *S. coelicolor*, the regulation of which was controlled by a sigma factor (σR).

The established Zur binding-sites from *Actinobacteria, Bacilli* and *Proteobacteria* are AT-rich palindromes found upstream from the genes being regulated [59; 62; 63]. Using a profile of Zur binding motifs from these bacterial groups, we searched the 26 genomes containing multiple copies of the S4 gene (from *Actinobacteria, Clostridia* and *Beta/Deltaproteobacteria*) for candidate Zur binding sites (see Methods). We were able to identify Zur binding sites upstream of the gene cluster of r-proteins L33, S14 and L28 and upstream of the paralogous genes of both L31 and L32 in *Salinispora arenicola* and *Salinispora tropica*, but no binding sites were found near the paralogous S4 genes. Neither were Zur binding sites found near ribosomal protein paralogs in the remaining genomes. Unfortunately, a Zur binding motif has not yet been reported for *Clostridia*, which comprises most of the genomes with paralogous copies of the S4 gene. Therefore, we can not exclude the possibility that the paralogous copies of S4 in *Clostridia* are regulated by Zur binding to a motif different from any in our profile.

However, according to gene expression data from two separate genomic-scale gene expression experiments in *C. acetobutylicum* [74] and *Clostridium novyi* [75], the paralogous C-(V) genes are not expressed under normal growth conditions but are up-regulated during sporulation. This leaves open the possibility that the C-(V) genes are related to some aspect of ribosomal function during sporulation and not used to regulate the zinc environment in *Clostridia*. If the C-(V) S4 proteins are indeed incorporated into ribosomes in clostridial spores, it would be interesting to examine any changes to these ribosomes, such as altered structure or changes in the assembly process.

3.3.3 Comparison of genome content near S4

Having used genome context in the analysis of several cases of horizontal transfer, we next examined the genome regions near S4 and the α-operon in the genomes of the bacteria without multiple copies looking for conserved patterns. Overall, the organization of the α-operon and nearby genes is highly conserved across a large number of bacterial groups. Many of the genomes have the conserved consensus gene cluster shown in Figure 3.6A, containing genes for initiation factor A (infA), L36, S13, S11, S4, RNA polymerase subunit A (rpoA) and L17. Variations are mainly seen in *Gammaproteobacteria* and *Magnetococcus*, which do not have infA near the cluster. Intriguingly, genes for both the C+ and C-(IV) forms of S4 can be found in the α-operon (green background in Figure 3.3). In fact, eight phyla have the gene for the C-(IV) form located in the α-operon, including all three of the phyla containing closely branching C+ and C-(IV)
Figure 3.6: Genomic content near S4 and the α-operon. Shown are a representative sample of the genomes. A) Consensus genome context of the S4 gene in the α-operon across most bacterial groups. B & C) Context of the α-operon and S4, respectively, in bacterial groups where an S4 gene is located outside the α-operon. Phyla abbreviations are given in parentheses. The lengths of the genes are to scale and gene are color-coded according to COG (clusters of orthologous groups) functional categories. (Figure courtesy of K Chen)

forms (Actinobacteria, Chloroflexi and Deinococcus-Thermus). Five other phyla, Aquificae, Bacteroidetes, Chlorobi, Fusobacteria, and Verrucomicrobia, contain only the C-(IV) form in the operon.

The remaining bacterial genomes, still covering a diverse set of bacteria, contain only an S4 gene of the C-(IV) form that is not located within the α-operon (red background in Figure 3.3). Figures 3.6B and C shows examples of the genomic context of the α-operon and the S4 gene, respectively, in these genomes. The organization of the genes remaining in the α-operon is unperturbed, but the context around the S4 gene is variable. Conservation of organization near the S4 gene can only be seen at the level of order or family; no correlations with the organization of any other genes could be detected at higher levels of taxonomy. When not located in the α-operon, the gene for S4 appears to be quite mobile.
3.4 Interpretation of the Evolutionary Patterns of S4

3.4.1 The ancestral form of S4 in the bacteria

Given the widespread occurrence of C-(IV) genes within the α-operon (see Figure 3.7), one must question the hypothesis that the C+ zinc-binding form of S4 is ancestral in the bacteria. If the C-(IV) form were a result of a single ancient gene duplication of a C+ gene, one would have expected to find nearly all of C-(IV) genes located outside of the α-operon. Instead, five classical bacteria phyla contain exclusively the C-(IV) gene in the α-operon. Moreover, three bacteria phyla contain monophyletic branches of both the...
C+ and C-(IV) genes, each organized in the typical α-operon style. Although it is known that horizontally transferred genes can replace their native copies in the genome, so called in situ gene displacement [76], such occurrences are still thought of as exceptions rather than the rule. The number of in situ displacements required to achieve the current distribution of C-(IV) genes in the α-operon would require replacement events of a much higher frequency or different character than that previously reported.

We propose instead that neither C+ nor C-(IV) is the sole ancestral form of S4 and interpret the data as implying the presence of both forms during the time when the bacterial lineages were diverging. The developing bacterial lineages would have sampled S4 genes from the bacterial pool according to some unknown criteria, perhaps related to the local environment (e.g., thermophilic organisms acquiring the zinc-binding form for added stability). While this sampling would have been functionally equivalent to HGT with in situ gene displacement, in that the gene order would be maintained, it would not have necessarily been mechanistically related to the process by which HGT occurs today.

Additional support for the existence of innovation sharing within gene pools comes from signatures in the S4 protein that were reported by Roberts et al. [46] to distinguish the bacterial and archaeal/eukaryal lineages. S4 proteins from both archaea and bacteria possess the RNA binding C-terminal domain, but have an N-terminal architecture distinct to each domain of life. Furthermore, the archaeal version of the α-operon is organized with S4 preceding S11 (S13-S4-S11), as opposed to S11 proceeding S4 as in bacteria (S13-S11-S4). Clearly, large-scale evolutionary changes occurred in S4 after (or at) the Bacteria and Archaea divergence, and yet the signatures are unvarying within each domain. Excluding the possibility that all extant bacteria can trace their vertical ancestry to a single individual cell and all extant archaea to another single cell, the respective organism pools at the time must have been able to efficiently share genes in an in situ manner that allowed the homogenization of the bacterial pool. This is the same evolutionary process required to support both a C+ and a C-(IV) form of the S4 gene in the bacterial pool.

### 3.4.2 Origin of S4 outside the α-operon

If, as suggested above, a bacterial pool allowed both the C+ and C-(IV) forms of the S4 gene to be brought into the genome in situ as needed, the question arises as to the origin of the C-(IV) gene outside of the α-operon in genomes where it is the sole copy. We propose that this organization is the result of HGT of the C-(IV) gene into C+ genomes after the phyla had diverged from the bacterial gene pool and the in situ evolutionary dynamic had slowed. Loss of the original C+ gene would have then allowed a reduction in zinc use without perturbation to growth of the organisms.

Figure 3.8 depicts the four possible evolutionary paths (labeled A–D) starting from either a C+ or C-(IV) gene inside the α-operon and ending with a single C-(IV) type gene outside the α-operon. Path A involves a gene duplication of a C+ type, mutation of the C+ type into a C-(IV) type, and finally loss of the original C+ gene. This path is ruled out for two reasons: first, no duplications of C+ S4 genes were observed in any of the 660 genomes studied, and second, the path depends on an unlikely set of mutation events. The C-(IV) genes outside the α-operon are indistinguishable in sequence from the C-(IV) genes inside, including the loss of a characteristic seven residue indel. The probability of an independent mutational deletion of seven residues from a C+ gene leading to the exact same indel pattern as in the pre-existing C-(IV) gene is low.
Additionally, there are other sequences signatures, such as residue 15 (in the RRXG motif) being glutamic acid in C+ and leucine/phenylalanine in C-(IV) and residue 21 being leucine in C+ and glycine/proline in C-(IV), that support a common origin for all of the C-(IV) sequences.

Path B specifies HGT of a C-(IV) gene into a genome containing the C+ type in the α-operon followed by loss of the C+ gene. Evidence supporting path B comes from the pattern of S4 HGT events presented in Results. The evolutionary history of S4 contains several relatively recent horizontal transfers, as supported by our analysis of genomes containing multiple copies of the S4 gene. In each of these cases, a C-(IV) gene was transferred into a genome with the C+ form of the S4 gene in the α-operon. Additionally, in two instances there was loss of the original C+ gene following the HGT of a C-(IV) gene, the clostridium F. magna and the betaproteobacterium M. flagellatus, exactly as prescribed in path B.

Both of the remaining paths, C and D, start with a C-(IV) gene and involve later acquisition of an additional C-(IV) gene either through duplication of the original or HGT, respectively. Our analysis found neither duplications nor horizontal transfers in any genome with the C-(IV) gene in the α-operon, although the sample size of known HGT events is low. Furthermore, there is phylogenetic evidence that some of the groups now containing only a C-(IV) gene outside of the α-operon are descended from lineages originally containing the C+ gene, which would preclude paths C and D. For example, all Alphaproteobacteria contain only the C-(IV) gene outside of the α-operon, while all other Proteobacteria have the C+ form (or recent variations thereof) in the α-operon. Even the genome of the unclassified proteobacterium Magnetococcus sp. MC-1, which is phylogenetically closest to the Alphaproteobacteria [77; 78], contains the gene for the C+ form of S4 in the α-operon. Thus, the Proteobacteria phylum likely contained the C+ gene originally.

The above arguments provide support for our hypothesis that C+ was the original form of the S4 gene in the branch of the tree containing C-(IV) outside the α-operon (red background in Figure 3.3) and that these branches received the C-(IV) gene through HGT. The original source of the C-(IV) S4 gene must have been one of the phyla containing C-(IV) natively, i.e., one with the C-(IV) gene in the α-operon, but once
Table 3.2: Taxonomic distributions of the C+/- ribosomal proteins

<table>
<thead>
<tr>
<th></th>
<th>L32</th>
<th>L36</th>
<th>L31</th>
<th>S14</th>
<th>L33</th>
<th>S18</th>
<th>L28</th>
<th>S4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Verrucomicrobiae</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>α-proteobacteria</td>
<td>+,−</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Bacteroidetes</td>
<td>+,−</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Chlamydiae</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Fusobacteria</td>
<td>+</td>
<td>nd</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>β-proteobacteria</td>
<td>+, +/−</td>
<td>+,−, +/−</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>+,−</td>
<td></td>
</tr>
<tr>
<td>γ-proteobacteria</td>
<td>-</td>
<td>+,−, +/−</td>
<td>+,−, +/−</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Chlorobi</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Planctomycetes</td>
<td>nd</td>
<td>-</td>
<td>+</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Deinococcus</td>
<td>+</td>
<td>+</td>
<td>−</td>
<td>+,−</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>,+,−</td>
</tr>
<tr>
<td>Cyanobacteria</td>
<td>−</td>
<td>+,−</td>
<td>+,−</td>
<td>+,−</td>
<td>+,−</td>
<td>-</td>
<td>-</td>
<td>+,−</td>
</tr>
<tr>
<td>Bacilli</td>
<td>+,−</td>
<td>+</td>
<td>−,+,−</td>
<td>+,−,+,−</td>
<td>−,+,−</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Tenericutes (Mollicutes)</td>
<td>+,−</td>
<td>+,−</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>−</td>
<td>−</td>
<td></td>
</tr>
<tr>
<td>ε-proteobacteria</td>
<td>−</td>
<td>+</td>
<td>+,−</td>
<td>+</td>
<td>+,−</td>
<td>+,−</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Actinobacteria</td>
<td>+,−</td>
<td>+</td>
<td>+,−, +/−</td>
<td>+,−, +/−</td>
<td>+,−, +/−</td>
<td>+,−, +/−</td>
<td>+,−, +/−</td>
<td>+,−</td>
</tr>
<tr>
<td>Chloroflexi</td>
<td>+</td>
<td>+</td>
<td>+,−</td>
<td>+</td>
<td>+,−</td>
<td>+,−, +/−</td>
<td>+,−, +/−</td>
<td>+,−</td>
</tr>
<tr>
<td>Spirochaetes</td>
<td>+</td>
<td>+</td>
<td>+,−</td>
<td>+</td>
<td>−</td>
<td>+,−</td>
<td>−</td>
<td>+,−</td>
</tr>
<tr>
<td>Thermus</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>−</td>
<td>−</td>
<td>+</td>
</tr>
<tr>
<td>Magnetococcus</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>−</td>
<td>−</td>
<td>+</td>
</tr>
<tr>
<td>Aquificae</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>−</td>
</tr>
<tr>
<td>δ-proteobacteria</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+,−</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>−</td>
</tr>
<tr>
<td>Clostridia</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+,+−</td>
</tr>
<tr>
<td>Acidobacteria</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Thermotogae</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
</tbody>
</table>

+,− indicates each form comprises at least 10% of the group, but occurs only once in each genome.
+/- indicates at least 10% of the genomes in the group contain both forms. This cut-off removes all minor cases of HGT.
nd indicates that the r-protein was not detected.

the S4 gene made the transition from an operon gene to a standalone gene it may have become more readily transferable. Later HGT events may therefore have originated from organisms having already received prior transfers. From the phylogenetic tree in Figure 3.3 the best candidate phyla for the original source are Aquificae, Bacteroidetes and Verrucomicrobia, but the low support values near the radiating points in the tree leave a great deal of ambiguity as to the exact source. Given the strict presence of the C-(IV) form and absence of an S4 gene in the α-operon in the groups, it appears that both the HGT events and native gene losses are likely ancient.

3.4.3 Evolutionary pressure and the loss of zinc binding in ribosomal proteins

Insight into an evolutionary process comes from not only describing the mechanism of change, but also the pressures behind the change. As discussed earlier, seven other r-proteins have been reported to bind zinc and to have evolutionary histories disrupted in a similar pattern to what we have reported for S4. Table 3.2 shows the occurrence of C+, C-, or both C+ and C- genes of these r-proteins in the major bacterial groups. C+ was reported as being the ancestral form of these r-proteins [58] and, if that is indeed the case, it is clear from the distribution that large groups have developed either the ability to do without zinc for specific r-proteins.
or to switch to C- paralogs under low zinc conditions. Specifically, we see characteristic divisions below the phyla level, *e.g.* Alphaproteobacteria have replaced the C+ genes for almost all zinc-binding *r*-proteins with C- genes, Epsilonproteobacteria has exclusively C- forms of three, and *Magnetococcus* sp. *MC-1* encodes the C- gene for only one.

These observations point to the conclusion that some bacteria evolved to use C- variants of the zinc-binding ribosomal proteins (including S4) to regulate the zinc economy of the cell. Whether this lower zinc usage was a response to a change in the zinc conditions in the environment or whether some other change in the environment (such as lower temperature) caused the decreased need for zinc is still unclear.

### 3.4.4 Conclusions

The cellular information processing system is generally believed to be much less subject to the influences of HGT than other genetic systems. While recent metagenomic studies have not reported any reliable HGT events for the ribosome among the three domains of life, examples of disagreement with the UPT among the bacterial versions of seven zinc-binding *r*-proteins S14, S18, L28, L31, L32, L33 and L36 have been well documented [56; 58]. According to our study of 660 bacterial genomes, the bacterial version of the universal *r*-protein S4, shares similarities with these seven proteins, namely they all have two different versions of the sequence, zinc-binding (C+) and non-zinc-binding (C-), and their evolutionary histories all show patterns of disagreement with the standard UPT.

The evolutionary history of *r*-protein S4 reconstructed here shows that S4 was subject to horizontal transfer throughout the history of the bacterial lineages. Recent HGT of the standard character was observed along with other less well-defined evolutionary dynamics of ancient origin. We propose wide-spread sampling of ancestral C+ and C- forms of the S4 gene from a bacterial gene pool as a possible explanation, but definitive proof of such an ancient event cannot be easily obtained. The present study was only possible given the large number of available bacterial genomes, and perhaps additional genomes of other diverse bacterial lineages would provide additional evidence for or against this proposition. Experiments detailing the purpose and regulation of paralogous S4 genes in *Clostridia* also may shed light on the differences between the C+ and C- forms.

In more practical terms, it should now be understood that even “core” proteins can have a more complex evolutionary history than can be explained by vertical inheritance. One recent study attempting to reconstruct an organismal tree of life included S4 in a concatenated gene tree [79]. Although the authors did attempt to remove genes subject to HGT, none was detected in the case of S4. It is clear from the present study that doing so is not always a simple proposition. Accurate evolutionary relationships for S4 were only uncovered with extensive coverage of the bacterial tree along with heavy use of genome content. Others have shown that concatenated genes trees may lack resolution [80], and this may be a direct result of mixing genes with different complex relationships, like the one reported here for S4.
3.5 Methods

3.5.1 S4 sequence acquisition and alignment

The analysis was based all of the complete bacterial genomes available at the time in NCBI GenBank. Additionally, in order to provide further data for a few poorly represented phyla, draft genomes from three *Planctomycetes*, five *Fusobacteria*, and six *Verrucomicrobia* were obtained from the Joint Genome Institute, as identified through the Genome OnLine Database (GOLD) [81].

To find S4 sequences and paralogs in the genomes, a non-redundant sequence profile was constructed as described in Sethi *et al.* [34] starting with annotated S4 sequences from the Swiss-Prot database [82]. This profile was used to do a BLAST search [83] on each genome with a cutoff of $10^{-7}$. Fragments containing only the C-terminal RNA binding domain were removed. Sequence were classified as C+ or C- by comparison to annotated sequences and then all sequences of each type were aligned using the ClustalW [84] multiple alignment function. The C+ and C- multiple alignments were combined using the ClustalW profile alignment function and the resulting alignment was hand edited to correct poorly aligned regions. All operations were performed within the MultiSeq [30] bioinformatics analysis environment.

3.5.2 Phylogenetic reconstructions

Maximum likelihood (ML) trees were reconstructed using RAxML version 7.0.4 [85]. A value of 10 was used for the maximum initial rearrangement distance (-i 10) and a value of 25 for the number of rate categories (-c 25). The tree for *Proteobacteria* was calculated using the JTT amino acid model [86] (-m PROTMIXJTT) and the tree for *Bacteria* using the WAG model [87] (-m PROTMIXWAG), as these models gave the best likelihood scores for a given maximum-parsimony tree of the respective alignments. A total of 1000 likelihood searches were performed for each alignment starting from unique, random maximum-parsimony trees (-f d -# 1000). The tree with the highest likelihood score was taken to be the ML tree. A consensus tree was constructed from the ML tree by removing bipartitions found in fewer than 50% of the other most likely trees. Following, 5000 non-parametric bootstrap runs were performed starting with the topology of the ML tree (-b -t ml.tre -# 5000) to determine support values for the bipartitions. Support values were mapped onto their corresponding branches in the consensus tree. Sequences from a few genomes (Candidatus *Carsonella ruddii* PV, *Sorangium cellulosum* ‘So ce 56’, *Symbiobacterium thermophilum* IAM 14863, *Petrotoga mobilis* SJ95, *Rubrobacter xylanophilus* DSM 9941, *Myxococcus xanthus* DK 1622, *Clostridium phytofermentans* ISDg) were highly mobile during ML reconstruction (likely long-branch artifacts) and so were excluded from the reconstruction and added afterwards using stepwise maximum-parsimony addition (-f p -t ml.tre).

3.5.3 Zinc regulatory motifs

Zinc regulation protein binding motifs, which are AT rich palindromes on the intergenic region of DNA strand were searched using MEME/MAST [88]. MEME was used to make a position specific substitution matrix (PSSM) based on input palindromes. The input profile of *Actinobacteria* was taken from experimen-
tally determined Zur binding sites in *Mycobacterium tuberculosis* and *Streptomyces coelicolor* [62; 63] and the profiles for *Bacillus* group and *γ*-proteobacteria were taken from [59]. Then the resulting matrices were used as input of MAST to search for other binding sites in the whole genomes. Only those genomes that have paralogs of S4 genes were subjected to this analysis.
Chapter 4

Unifying Sequence and Structure Data for Evolutionary Analysis

4.1 Background

In the field of bioinformatics, research activities are often split into two distinct areas: sequence analysis or structure analysis. Genomic and other sequencing projects generate enormous amounts of sequence data that are initially released with large portions annotated as either putative or hypothetical. Structural data, even in the era of structural genomics, are produced at a slower pace but analyzed to a high degree before being deposited into the public databases, such as PDB [89], SCOP [90; 91]/Astral [92], CATH [93] and NDB [94]. This difference in pace has led to an increasing discrepancy in the relative sizes of these two data sets. The total size of the sequence databases (NCBI [95; 96], EMBL [97], DDBJ [98], JGI [99], RDP [100], Swiss-Prot/TrEMBL [101], CRW [102], Bayreuth tRNA compilations [103], and the Genomic tRNA Database [104]) is several orders of magnitude greater than that of the structure databases. For a given protein, the large number of available sequences allows more complete evolutionary analyses. Multiple sequence alignments (MSA) are instrumental in identifying key conserved areas of a sequence, developing an evolutionary history of a molecule [105], and examining the covariance within a sequence in response to evolutionary pressure [106]. These analyses depend on having enough sequences to perform a well-balanced statistical analysis. The advantages of structural data are that they provide much more detailed information about the molecule in question, allowing specific atomic level interactions to be analyzed. Additionally, since structure is more conserved than sequence [107], structural data can be used to reconstruct many of the deeper evolutionary branches that would be difficult or impossible to determine with sequence data alone [10; 11; 34; 108].

VMD [109], which currently has more than 30,000 registered users, provides powerful visualization and analysis capabilities for both structural and dynamics data generated from molecular dynamics simulations, as well as energetics derived from molecular mechanics force fields. It is optimized to handle large scale systems containing millions of atoms. VMD also implements a flexible scripting interface for the creation

The contents of this chapter are based in part on work previously published as Elijah Roberts, John Eargle, Dan Wright, and Zaida Luthey-Schulten. “MultiSeq: unifying sequence and structure data for evolutionary analysis,” BMC Bioinformatics, 7:382 (2006). Additionally, the work on determining a QR cutoff using a maximum-entropy criteria was performed by Elijah Roberts, Jonathan Montoya, Evan Rosenfeld, and Zaida Luthey-Schulten.
of custom tools. The previous Multiple Alignment [110] extension to VMD added only the capability to use evolutionary information obtained from multiple structures for interpreting structural results. Our goal with MultiSeq is to extend VMD’s capabilities further by incorporating the more diverse evolutionary data available in sequences into the analysis process.

There are already a large number of tools available to analyze bioinformatics data, but, like the field itself, they are mostly segregated into either sequence or structure tools. In the sequence world there are tools for viewing, analyzing, and editing an MSA like AE2 [111], CINEMA [112], ClustalX [113], and Jalview [114]; there are tools for creating MSAs by aligning individual sequences and profiles, like ClustalW [115], HMMER [116] and T-Coffee [117]; there are tools for annotating sequence data such as Pfaat [118]; and BLAST [119] is used for searching through databases for related sequences. One popular package, MEGA3 [120], provides an evolutionary approach to analyzing protein and nucleic acid sequences, including many easy to use features for determining sequence based phylogenies. Similarly, in the structure world there are numerous tools for visualizing structural data and performing structural analyses, including RASMOL [121], STAMP [122], STRIDE [123], and 3DNA [124].

There are also a few programs that combine sequence and structure data either for specific purposes, such as Swiss-PdbViewer/SWISS-MODEL [125] and MolIDE/SCWRL3 [126; 127] for homology modeling, or as part of a pre-computed database of attributes, of which STING [128] is the primary example. Modeler [129] allows structural features to be built using sequences and structures of homologous proteins and nucleic acids. UCSF Chimera [130], a well known molecular modeling program that originated to handle small molecule docking, provides the ability to use sequence data in conjunction with structural data. However, it lacks some of the features needed to perform well-balanced evolutionary analyses, such as phylogenetic tree construction and elimination of bias. Friend [131], a bioinformatics application, has many of the sequence features required for performing evolutionary analyses, but has insufficient structural functionality to fully interpret the results in a structural context. InsightII and Discovery Studio (Accelrys Inc.) and MOE (Chemical Computing Group Inc.) are popular commercial packages for analyzing protein/drug interactions based on protein structure, dynamics, and energetics. Both programs can also use sequence data to perform combined analyses. NCBI’s Cn3D [132] also supports both sequence and structure data, although it is primarily designed for use with pre-computed 3D superpositions and MSAs.

Here we present MultiSeq, a unified bioinformatics analysis environment that allows one to organize, display, align and analyze both sequence and structure data for proteins and nucleic acids. While special emphasis is placed on analyzing the data within the framework of evolutionary biology, the environment is also flexible enough to accommodate other usage patterns. The evolutionary approach is supported by the use of predefined metadata, adherence to standard ontological mappings, and the ability for the user to adjust these classifications using an electronic notebook. MultiSeq contains a new algorithm to generate complete evolutionary profiles that represent the topology of the molecular phylogenetic tree of a homologous group of distantly related proteins. The method, based on the multidimensional QR factorization of multiple sequence and structure alignments, removes redundancy from the alignments and orders the protein sequences by increasing linear dependence, resulting in the identification of a minimal basis set of sequences that spans the evolutionary space of the homologous group of proteins.
4.2 MultiSeq Software Description

4.2.1 Unification of sequence and structural data in evolutionary profiles

The complementary information provided by fusion of sequence and structural data sources can give insight into evolutionary changes in sequence, structure, and function. However, the conceptual spaces of these fields differ, often resulting in mutual incomprehensibility to researchers in each field. MultiSeq, in dealing with both sequence and structure data in a way that is accessible to both areas, helps to bridge this gap. It does so through the use of integrated cross-referencing that acts as an informal version of ontology-driven knowledge extraction and discovery [133]. We plan to enhance future versions of MultiSeq to incorporate formal ontological methods, including using the work of groups such as the Gene Ontology project (http://www.geneontology.org/).

There is tremendous utility to be had in combining both sequence and structure data within an evolutionary framework using the four pillars of information science, information visualization, mathematics, and biology to organize the flow of information. An evolutionary profile (EP) is a concise and complete representation of the diversity that has been generated by the evolutionary process within a homologous group of proteins. A key step in the creation of an EP is the elimination of redundancy present in the sequence and structural databases [134] due to bias in the selection of organisms chosen for study. The sequence and structure QR algorithms have been developed specifically to address this problem [11; 34]. These smaller, more evolutionarily balanced profiles have comparable, and in many cases better, performance in database searches than conventional profiles containing hundreds of sequences. For more diverse families or superfamilies, with sequence identity < 30%, structural alignments, based purely on the geometry of the protein structures, provide better alignments than pure sequence-based methods. Merging the structure and sequence information allows the construction of accurate profiles for distantly related groups. The success of using sequence and structure based EPs for both gene annotation [108] and the prediction of structurally conserved motifs [135] shows their effectiveness. We also anticipate the usefulness of EPs for studying, among other things, the relationship between protein structure and stability, the evolution of protein/RNA interfaces, and the basis of protein conformational motion.

The actual process of creating an EP is detailed in Sethi et. al. [34], a tutorial [136], and a forthcoming applications paper, but can be summarized as follows:

1. Load a set of sequences and structures and their associated metadata.
2. Align the data, using structural alignments as profiles for aligning widely divergent sequence groups.
3. Perform a phylogenetic analysis to determine the evolutionary relationships in the data.
4. Check and adjust the alignment using the phylogenetic tree and taxonomic information as guides.
5. Eliminate any redundant data that may be a source of bias.

Within this process there are many difficulties, such as identifying horizontal gene transfer (HGT) events and misannotated data, both important for proper grouping of evolutionary data, and developing a statistically well-balanced set of sequences and structures. MultiSeq attempts to lower some of these barriers...
Figure 4.1: Overview of the MultiSeq environment showing aligned sequence and structural data. (1) 1D representation of structural data colored by structural conservation. (2) 1D representation of sequence data colored by sequence identity. (3) 3D representation of structural data colored by structural conservation, as shown by VMD. For structural data, the coloring is synchronized between the 1D representation and the 3D representation.

to combining sequence and structural data into EPs by consolidating the tools necessary to perform such analyses in an intuitive software package (Figure 4.1).

4.2.2 Importing protein data for analysis

The primary function of MultiSeq is to provide an environment for the evolutionary analysis of bioinformatics data from both structure and sequence. Before any analysis can be performed, however, the data must first be imported into the environment, which is often a non-trivial task given the wide variety of sources from which data may be acquired. MultiSeq provides a consistent interface to allow data from numerous sources to be quickly and easily brought into the environment and consolidated for further analysis. Structural data for biomolecules come in a bewildering array of file formats, a large number of which can be read and processed by VMD. To take advantage of this capability, MultiSeq relies on VMD to parse structure files and present a 3D representation of the data. After VMD has loaded the structural data, MultiSeq creates a copy of the sequence portion of the data, stores that in its own internal data structures for use when displaying 1D representations of the data (see Figure 4.1), and then establishes a link between its internal data structures and those of VMD. This synergy means that MultiSeq works with every format of structural data that VMD supports, including such common formats as PDB, XYZ, NetCDF, and CHARMM. MultiSeq also makes it easy to load multiple structures, which is necessary during the construction of a
structural profile. Additionally, MultiSeq extends VMD’s ability to load protein structures over the Internet by allowing multiple PDB codes to be specified and individual domains of protein structures to be loaded directly from the Astral database [92]. Sequence data are often stored in a single file containing multiple sequences, in either an aligned or an unaligned state. MultiSeq can load sequence files formatted in ALN, FASTA, Nexus, PIR, and PHY file formats. For FASTA formatted files, description lines are preserved and made available through the electronic notebook, described below. Upon loading a sequence file, MultiSeq can automatically download corresponding structural data, if it is available in one of the known structural databases. Currently supported structural databases are the PDB, Astral, and the subset of Swiss-Prot that is derived from the PDB.

A final method of loading protein sequence data into MultiSeq is through the use of a BLASTP [137] search. Given a target sequence or profile, BLAST discovers a variety of homologous sequences which can then be incorporated into the analysis. MultiSeq uses a locally installed version of BLAST to search local sequence databases using a single sequence, a profile of sequences [119], or a fragment of a sequence or profile. The search can be performed a single time or iteratively using PSI-BLAST, and the search results are displayed and filtered before being imported into MultiSeq, as shown in Figure 4.2. Current filtering options include BLAST e-score, taxonomic classification, and a redundancy filter based on the sequence QR algorithm [34] on the BLAST generated alignment. As when loading sequence data from a file, any corresponding structural data for the search results can be automatically downloaded when the search results are imported. This opens up the possibility of running a BLAST search against the PDB or Astral databases to load structures that share sequence similarity with a source sequence or profile, a feature that is particularly useful for finding a template during homology modeling of a protein of unknown structure [135]. Depending on the size of the protein, MultiSeq can easily load hundreds of sequences and/or structures. The time required to perform an analysis of such a large set, however, depends on the analysis method being used.
4.2.3 Organizing data to accommodate various analysis frameworks

The number of different sources of sequences and structures can be intimidating and calls for an organizational framework in which to work with the data. At the same time, the varied uses of these data demand that the framework be flexible enough to accommodate a wide variety of users. MultiSeq addresses this issue by implementing a flexible grouping system. Each sequence in MultiSeq is displayed beneath its group in the main display, as shown in Figure 4.3A. The group header acts as an interface anchor to allow the user to perform operations on the group as a whole, and the status bar shows overview information about the currently selected group. The default grouping is based on the source of the data, i.e., structures loaded through VMD appear in the VMD Structures group and sequences loaded by a BLAST search appear in the BLAST Results group, but the user can easily expand, rename, and reorder these groupings as appropriate for the situation at hand. Additionally, the data can be automatically grouped by taxonomic classifications (Figure 4.3B). Separating the data into evolutionarily distinct groupings allows any analysis to be easily performed on each related group independently.

4.2.4 Finding metadata automatically via the Internet

Metadata (or “data about data”) – such as taxonomy, enzymatic function, or structural classification – related to sequence and structural data can provide valuable insight during many bioinformatics analyses. Various databases accessible via the Internet store this information and present it when displaying results but otherwise make little use of it. MultiSeq correlates this metadata by cross-referencing both the name of the sequence or structure and any source information contained in the original file. Currently, MultiSeq can
Figure 4.4: MultiSeq tools. (1) The electronic notebook displays various metadata associated with the sequence and also provides space for making annotations about a sequence. Changes will be saved in the MultiSeq session. (2) The phylogenetic tree viewer shows evolutionary relationships amongst the data. Data are labeled by species name and colored by domain of life, those highlighted in yellow are part of the selected non-redundant set. (3) The QR ordering of the non-redundant set is also displayed, lower numbers indicate data that are more linearly independent. (4) The plotter allows a metric to be plotted along the length (or a subset) of the sequence. All of the coloring metrics can also be used by the plotter.

extract NCBI taxonomy information [138], Enzyme Commission (EC) numbers, and SCOP structural classifications [90]. MultiSeq integrates this metadata into the evolutionary analysis process through grouping and phylogenetic tree functions. Metadata can be added, viewed, and edited using the electronic notebook (Figure 4.4). The electronic notebook provides a consistent way to interact with all available metadata for a sequence, regardless of its source. It also provides a place to store notes regarding the sequence and any processing that has been performed on it. Changes to the metadata are saved along with a MultiSeq session, described below.

4.2.5 Alignment of sequence and structural data

In order to properly analyze multiple homologous sequences and structures, they must first be aligned. For structural data, a version of STAMP [122] that has been modified to better align end regions (details of the modifications are available in the methods section) is used to perform the alignment. For sequence data ClustalW [115] is used. In the next version of MultiSeq, a plug-in framework is planned to allow other sequence and structural alignment programs, such as HMMER [116] and T-Coffee [117], to be used. It is generally accepted that structural alignments are more reliable than sequence alignments for distantly related proteins and RNA molecules [107; 139; 140], so MultiSeq allows a structural alignment to be passed to ClustalW as a profile to seed the alignment process. This technique can be particularly effective when the structural profile is in fact an evolutionary profile and contains a non-redundant sampling of distantly related structures. In either case, though, the alignment is rarely perfect and some manual editing is usually required using MultiSeq’s built-in editor.
4.2.6 QR algorithms to eliminate redundancy and bias in data

Although the vast quantity of data available in this post-genomic era brings many new possibilities for analysis, it also opens up the potential for introducing systematic errors in these analyses due to the biases inherent in the makeup of the various databases. MultiSeq includes both the sequence QR [34] and structure QR [11] algorithms to help detect and eliminate this redundancy during any step of the analysis process. The sequence and structure QR algorithms orthogonally encode a multiple sequence or structure alignment as a multidimensional matrix, and then perform a QR factorization on this matrix [141]. The result is an ordering of the sequences or structures from most linearly independent to least independent. A non-redundant set from amongst the available data (See Figure 4.4) is constructed by specifying a cutoff in either sequence or structural similarity. The QR algorithms can also be run on a specific region of the MSA so that the non-redundant set can be generated based on, for example, one domain of a multi-domain protein or an insertion in the sequence. The sequence and structure QR algorithms combined with the grouping and selection capabilities of MultiSeq constitute a powerful environment for constructing EPs for use in bioinformatics-intensive tasks such as homology modeling [135] or gene annotation [34; 108].

4.2.7 Analyzing phylogenetic relationships

Phylogenetic trees, which show the relationships between related proteins or nucleic acids, are invaluable when performing evolutionary analyses. They provide a guide for investigating why and how certain attributes developed as well as identifying misalignments and HGT events. The accuracy and speed of various tree reconstruction methods, however, varies widely from simple distance based methods such as un-weighted pair group method with arithmetic averages (UPGMA) [142] and neighbor-joining (NJ) to complex methods such as maximum likelihood [143], which take into account an underlying theory of evolution. In general, distance based trees are sufficient for many common uses [144]. MultiSeq creates UPGMA trees using the structural measures $Q_H$ [10] and root mean square deviation (RMSD) as well as the sequence measure of percent identity. It also creates trees based on similarity using the NJ method of CLUSTALW [115]. After a tree has been computed, it can be decorated and colored with various attributes such as species name, domain of life, and enzymatic function, as shown in Figure 4.4. Additionally, various manipulations such as collapsing, rotating, and labeling nodes to assist in visualization can be performed. One further use for phylogenetic trees within MultiSeq is in conjunction with the QR algorithms to eliminate redundancy from data. When either the SeqQR or StructQR tools are used on data being displayed in a phylogenetic tree, those data are highlighted both within the main environment and within the tree viewer (Figure 4.4). This feature allows for evaluation of the non-redundant selection so that the user can adjust the cutoff. The orderings from the QR algorithm indicating which data are most linearly independent are also displayed in the tree to assist in this process.

4.2.8 Using visualization to illuminate trends

One way to effectively present complex information is to color code the data using attributes that are not normally visible [145]. MultiSeq presets attributes of the data as coloring in both the 1D representation of
the sequence portion of the data and the 3D representation of the structural portion of the data. It maintains a consistent coloring between the two representations in order to facilitate an easy mental transition between them. Many different sequence and structural metrics are currently implemented as coloring choices and the addition of custom coloring methods is supported through a programming interface. The current list of standard metrics is: sequence conservation, sequence entropy, percent sequence identity, sequence similarity, \( Q_{res} \) structural similarity, residue type, and structural RMSD. In addition to calculating attribute values, MultiSeq can import them from a tab or space delimited file. This enables the importing of other types of attribute data, such as those from HD exchange or \( \Phi \)-value experiments. Many of the above coloring metrics are calculated by comparing two or more sequences or structures to get a value representing the specified attribute. The default behavior is to use all of the loaded data in the calculation of a metric, however, one can optionally have MultiSeq process each group independently. Using this feature one can, for example, view sequence identity across all domains of life and then quickly switch to sequence identity within the individual domains. This capability can be very useful for identifying a signature of a specific group of sequences or structures. Another method of assisting in visualization is to hide attributes that provide no relevant information in the current context. Often, eliminating this extraneous information can lead to patterns being more quickly understood. One common technique of dealing with this issue in the world of structural biology is through the use of secondary structure representations, showing only the backbone and secondary structure elements of a protein. VMD supports secondary structure, as calculated by STRIDE [123], as a 3D representation and MultiSeq can display it for structures as a graphical 1D representation. MultiSeq also provides a bar and line representation that is particularly useful when visualizing attributes that are zero over portions of the sequence, such as experimental data.

4.2.9 Nucleic acid sequences and structures

MultiSeq also supports bioinformatics analysis of both nucleic acid sequence and structure data, but the tools are somewhat more limited in the present release. Nucleic acid sequences may be imported as unaligned sequences or as MSAs using any of file formats supported for protein sequences. These data may be obtained from a variety of databases including IMG, NCBI (Genbank), Bayreuth tRNA compilations, CRW, RDP, and the Genomic tRNA Database. We provide external scripts to convert files from AE2 format (provided by Gary Olsen) and Bayreuth flatfiles to the FASTA format (http://www.scs.uiuc.edu/schulten/software.html). BLASTN support for finding related nucleic acid sequences is planned for the next release. Once nucleic acid sequence data have been loaded, multiple sequence alignments can be computed using the ClustalW interface within MultiSeq. Only coloring by sequence identity works with nucleic acids, other sequence-based coloring metrics specific for nucleic acid will be incorporated in the next release. STAMP has been modified to align nucleic acid structures by their backbone phosphorous atoms resulting in a structural alignment analogous to \( \alpha \)-carbon based alignment for proteins. When the alignment is complete the 3D representation displays the structural superposition of the aligned molecules. The built-in structural alignment analysis tools, such as structure-based trees and coloring metrics, work correctly with nucleic acid structural alignments. RNA molecules frequently incorporate nonstandard modified nucleotides that can affect folding, structure, and function. For example, the T\( \psi \)/C loop in tRNA typically contains a \( \psi \), or pseudouridine,
base. There are on the order of 100 RNA-associated modified bases identified at this time [146]. The RNA molecule, as opposed to its DNA gene, must be sequenced to determine the modified bases included. When this information is available in structure or sequence files MultiSeq recognizes and appropriately displays modified bases in the 1D representation. In the next release of MultiSeq, QR will be available for nucleic acids, and a canonical, evolutionarily balanced 16S rRNA will be incorporated to help with phylogenetic analysis. At that time secondary structure analysis tools for nucleic acid structures will also be included.

4.2.10 Exporting data

It is often desirable to preserve an entire bioinformatics analysis so that work can be resumed at a later time. MultiSeq implements this by saving the entire environment as a session. When a session is saved, all of the sequence and structure data loaded into VMD and MultiSeq are saved along with any alignments and transformations that have been applied to them. Metrics, annotations, metadata changes, and representation choices are also saved with the session. Once a session has been saved, it can later be loaded and work resumed quickly and easily. MultiSeq also supports numerous formats for exporting all or a subset of the data in the environment to a file. This can be useful if an analysis needs to be run using external bioinformatics software. For example, MultiSeq can export all of the files necessary to run a maximum likelihood/parsimony based phylogenetic analysis of sequence data using PAUP* [147], PHYLIP [148], and PHYML [149] or a Bayesian based analysis using MrBayes [150]. A final feature of note is MultiSeq’s ability to export publication quality graphics. The sequence window, tree viewer, and plotter can all save PostScript files of their current representation. Since these are vector graphics, they can be scaled and manipulated using illustration software with no loss of quality.

4.2.11 Methods

$Q_{res}$

We use a measure called $Q_{res}$ to calculate structural similarity of each residue in a set of aligned structures. It is derived from $Q$, which is used in protein folding to compare the pair distances in a protein conformation to the native one [151]. We have previously used this measure for deriving protein cores by looking at structural conservation [10; 11]. $Q_{res}$ computes the similarity of the $C_\alpha$-$C_\alpha$ distances between a residue and all other residues in the protein, excluding nearest neighbors, to the corresponding distances in a given set of proteins. The result is a value between 0 and 1 that describes the similarity of the structural environment of a residue in a particular protein to the environment of that same residue in all other proteins in the set. Lower scores indicate low similarity and higher scores high similarity. Formally, $Q_{res}$ is defined as follows:

$$Q^{(i,n)}_{res} = \frac{1}{\text{proteins}} \sum_{(m\neq n)} \frac{\text{residues}}{j \neq i-1,i,i+1} \exp \left[ -\frac{\left(r_{ij}^{(n)} - r_{ij'}^{(m)}\right)^2}{2\sigma_{ij}^2} \right]$$ (4.1)

where $Q_{res}^{(i,n)}$ is the structural similarity of the $i^{th}$ residue in the $n^{th}$ protein, $r_{ij}^{(n)}$ is the $C_\alpha$-$C_\alpha$ distance between residues $i$ and $j$ in protein $n$ and $r_{ij'}^{(m)}$ is the $C_\alpha$-$C_\alpha$ distance between residues $i'$ and $j'$ in protein $m$. 

49
that correspond to residues $i$ and $j$ in protein $n$. The variance is related to the sequence separation between residues $i$ and $j$, $\sigma_{ij}^2 = |i - j|^{0.15}$, and the normalization is $\mathcal{N} = ((N_{\text{seq}} - 1)(N_{\text{res}} - k))^{-1}$, where $N_{\text{seq}}$ is the number of proteins in the set, $N_{\text{res}}$ is the number of residues in protein $n$, and $k = 3$ except when residue $i$ is the N or C-terminus in which case $k = 2$.

$Q_H$

For measuring the similarity between two structures, we use $Q_H$, which we have previously derived [10; 11]. Like $Q_{\text{res}}$, it is also adapted from $Q$, but accounts for the presence of insertions in the structure. Briefly, $Q_H$ calculates an overall score for the similarity of two structures by summing the similarity of all residues and then adding a term for each gap in the alignment. The more that an insertion perturbs the structure of nearby regions, the lower the resulting $Q_H$ value.

**QR factorization**

The sequence and structure QR algorithms eliminate the redundancy from a collection of sequences or structures, respectively. The output is the smallest set of sequences or structures that represents the evolutionary diversity present in the initial group. These algorithms are based on a QR factorization with column pivoting of a matrix encoding the sequence or structure alignment. We have described each of these algorithms and their utility in developing EPs previously [11; 34].

**STAMP**

The STAMP structural alignment program generates both structural superpositions and sequence alignments using tertiary structure comparisons [122]. Two modifications were made to the STAMP structural alignment program included with MultiSeq. First, the program was modified to work with RNA and DNA by allowing it to read structure files containing the phosphate backbone atoms of nucleic acid molecule and to recognize the residues contained in these files. Second, the program was modified to insert gaps into the multiple sequence alignment so that the trailing, poorly aligned ends of different structures will be gapped with respect to one another. These end-gaps are a natural result of the dynamic programming local alignment algorithm used by STAMP.

**C++ bioinformatics library**

Many of the algorithms are written in C++, since TCL is less suited for computationally intensive work. To facilitate the development and implementation of these algorithms, we have developed *libbiokit*, a bioinformatics toolkit. This library is comprised of classes that perform file I/O, such as FASTA and PDB readers and writers; classes that represent commonly used bioinformatics data structures, like sequence and structure alignments; and stand-alone utilities that execute the QR, $Q_H$, $Q_{\text{res}}$, and phylogenetic algorithms along with other standard measures used in the analysis of bioinformatics data. Libbiokit is packaged with MultiSeq and is also available separately as open source software from our website (http://www.scs.uiuc.edu/schulten/software.html).
4.2.12 Concluding remarks

MultiSeq allows new approaches to be taken in bioinformatics analysis: new relationships can be found and investigated by combining sequence and structure data; automatic download and use of metadata along with flexible grouping encourages organized analysis of unfamiliar data; the ability to remove redundancy from large sets of data helps to focus and speed up evolutionary analyses; and integration with several popular bioinformatics tools along with a versatile input and output ability reduce the time and “busy work” overhead of performing any analysis. MultiSeq extends VMD’s capabilities into the realm of sequences based data and we hope that MultiSeq will help bring more widespread use of sequence data to the world of structural biology and vice versa. MultiSeq is included with VMD starting with version 1.8.5 (http://www.ks.uiuc.edu/Research/vmd). MultiSeq benefits from VMD’s cross platform nature and currently runs on numerous operating systems, including Linux, Mac OS X, Solaris, and Windows. Metadata databases are automatically downloaded and updated via the Internet and can be stored either on the user’s local machine or a workgroup file server. The use of BLAST for searching requires a locally installed version of the BLAST software from NCBI and sequence databases stored either on the local machine or a workgroup server. Detailed instructions on configuring the software are available in the MultiSeq manual available online (http://www.scs.uiuc.edu/schulten/multiseq). A tutorial is also available from the NIH Resource for Macromolecular Modeling and Bioinformatics to assist in learning how to use the features of MultiSeq described in this article [136].

4.3 Unbiased Evolutionary Profiles Using Shannon Entropy

4.3.1 Bias in sequence databases

Many combined pressures are causing the rapidly growing biological sequence databases to be over-populated by certain species and not representative of the sequence diversity explored by the evolutionary process [152; 153]. Some of this over-representation is artificial, occurring because of investigator preference in sequencing well-known, disease related, or model systems. Some is technological, due to limitations in sequencing and sampling techniques. Some is natural, resulting from the normal variations in speciation rates among different lineages. In a phylogenetic tree, over-representation reveals itself in the shape of the tree, i.e., variation in the number of taxa descended along different branches. This shape variation is a measure of the imbalance of the phylogeny underlying the tree, where “balance” is defined by agreement with a particular evolutionary model [154].

The universal phylogenetic tree (UPT; the Tree of Life), describing our current view of the evolution of the organismal lineages, represents an unbalanced phylogeny [155; 156]. Even if all extant species were known, differences in speciation or mutation rates would cause the appearance of many more species along some branches of the UPT than others, leading to imbalance. When performing an evolutionary analysis using biological sequence data, it is important to make the distinction that sequence space (the space spanned by all orthologous sequences) and organismal space (the set of all species) do not necessarily coincide. While each species can be considered a distinct entity, sequences from related species have overlap with each other. If one simply selected a sequence from each possible species, the set would be dominated
by the many similar sequences from the quickly speciating lineages. Potentially worse, if one only used a sequence from each identified species the set would be dominated by the biased sampling of species that is currently available from sequencing efforts. What one often wants for a sequence analysis is a set that is representative of the sequence space (known as a profile) and not the organismal space. Unless taken into account, over-representation in a database or a sequence alignment becomes bias in a profile.

Bioinformatics methods analyze the available sequence data to search for patterns and allow hypotheses to be tested. While it might be assumed that more data is necessarily better for these methods, many do not incorporate an evolutionary model through which correlations in the data due to phylogenetic relationships can be excluded from the analysis [157]. Neither do they implement sequence weighting techniques by which over-representation can be compensated [158]. These methods can therefore only properly be used with data of a representative composition (a profile). For example, hidden Markov models are probabilistic models commonly used in biology for searching through sequence databases and for aligning sequences to profiles [159]. Being probabilistic, these models require a known data set with which they can be “trained” to recognize specific patterns. Over-representation of some sequences in the training set leads to the inflation of some probabilities in the model and to poor performance when analyzing sequences similar to those underrepresented in the training set [160; 161]. As a consequence, most hidden Markov models attempt to discriminate against redundancy during the training phase, though not typically by using evolutionary models. Analysis of sequence conservation, typically used to infer the presence of a constraint on the evolution of a sequence (e.g., an active site), is also affected by the selection of sequences for analysis. Over-representation can artificially inflate the conservation value of certain residues, making them appear more constrained than if the sequences being analyzed were representative [162; 163].

Even methods where the addition of closely related data can be helpful, such as phylogenetic reconstruction [164; 165], have limitations regarding the size of the data sets they can analyze. If the data must be curated to a certain size before the analysis begins, it is best to start with a set that samples the desired level of taxonomic diversity in a non-biased fashion and then add additional data to increase the sampling density where needed [166]. Although one must always be cautious of long-branch attraction and node-density artifacts in sparse phylogenetic reconstructions.

To address the problem of over-representation in the available sequence data, Sethi et al. [34] introduced the sequence QR method for removing redundancy from molecular sequence data based on earlier studies of redundancy in protein structures [10; 11]. The method represents a sequence alignment as an orthogonally encoded matrix of vectors in a multi-dimensional sequence space, allowing redundancy to be expressed as linear dependence of the sequence vectors and diversity as degree of linear independence. It uses the QR factorization of this matrix to obtain an ordering of the sequences in terms of increasing linear dependence. The ordering starts with the sequence that is (on average) the most different from the others in the alignment and each following sequence is the one that is most distinct from all of those coming before it (see 4.3.4). Selecting sequences according to this ordering until the maximum percent identity of any pair in the set exceeds a specified cutoff value results in a set termed an “evolutionary profile” (EP) because, for the specified cutoff, it best represents the evolutionary sequence diversity encompassed by the full set. The sequence QR method has been used extensively to generate statistically unbiased profiles of protein sequences and
structures from a wide variety of functional classes [34](supporting information). EPs have been shown
to capture the sequence diversity needed for database searching with a much smaller number of sequences
than other types of commonly used profiles [34; 108]. By incorporating evolutionary (phylogenetic) infor-
mation into a profile through the selection of sequences, the sensitivity of bioinformatics methods that do
not explicitly use such information can be increased.

Despite their successes, EPs still require a manual determination of the cutoff value that is appropriate for
the analysis method. For database searching, a cutoff value chosen arbitrarily in the range of 50–75% iden-
tity is almost always sufficient for profile search algorithms such as PSI-BLAST [167] and HMMER [159].
When generating EPs for more general purposes, however, what is truly desired is an automatic determi-
nation of the cutoff based on the information content of the sequences. In order to construct an unbiased
profile of sequences, one would ideally select sequences in order of decreasing diversity until the diversity
gained from the last added sequence exactly equaled the increase in redundancy. Any additional sequences
added past this cutoff would only increase the overall redundancy of the profile. We propose that such a
cutoff criteria is met when the information (Shannon) entropy of the profile is at a maximum.

4.3.2 Information entropy of a multiple sequence alignment

The information entropy, \( H(X) \), of a discrete random variable, \( X \), is a measure of uncertainty about its
value and is the cornerstone of the theory of information founded by Claude Shannon [168]. If \( p(i) \) is the
probability of \( X \) being in state \( i \) out of possible states \( 1 \) to \( D \), then

\[
H(X) = - \sum_{i=1}^{D} p(i) \log_2 p(i).
\]

If the logarithm is base 2, then \( H(X) \) is measured in bits and can be thought of as the average number of
yes or no questions required to discern the state of \( X \). The entropy (uncertainty) is at a maximum when the
probability of being in each state is equal. For a sequence of \( L \) independent random variables \( X_1 \) to \( X_L \), the
total entropy of the sequence is the sum of the entropy of each individual variable:

\[
H_{\text{seq}} = \sum_{j=1}^{L} H(X_j),
\]

\[
= - \sum_{i=1}^{D} \sum_{j=1}^{L} p(i, j) \log_2 p(i, j),
\]

(4.2)

where \( p(i, j) \) is the probability of \( X_j \) being in state \( i \).

For a single biological sequence, obtaining the necessary probabilities for various residues at each po-
sition is difficult. However, given a multiple sequence alignment of many homologous sequences, it is
possible to estimate the probability of a residue occurring at a specific position by dividing the number of
occurrences of the residue at that position in the alignment by the total number of sequences. Using these
probability estimates, defining \( 0 \cdot \log_2 0 \) to be equal to 0, and assuming that the positions are independent,
Equation (4.2) can be used to calculate the entropy of an alignment of \( L \) homologous residues in \( N \) aligned sequences:

\[
H_{\text{aln}} = - \sum_{j=1}^{L} \sum_{i=1}^{D} \frac{\text{freq}(i,j)}{N} \log_2 \left( \frac{\text{freq}(i,j)}{N} \right),
\]

where \( D \) is the total number of characters in the alphabet (20 for proteins and 4 for nucleic acids) and \( \text{freq}(i,j) \) is the number of occurrences of residue type \( i \) at position \( j \) in the alignment. This method has been successfully used to analyze sequences for conservation [169], binding sites [170; 171], and other factors (see Adami [172] for a review).

### 4.3.3 Maximum entropy profile to determine the QR cutoff

Considering sequence entropy in regard to the issue of choosing a QR cutoff, one can see that expanding an alignment by adding a sequence will increase its entropy if, on average, the entropy of each position goes up. This will be the case if the sequence contains a preponderance of residues that are different from those already present at their positions in the alignment, i.e., if it increases the diversity of the alignment. On the other hand, if the sequence is similar to ones already in the alignment, the entropy of each position will tend to decrease, lowering the overall entropy. The principle of the maximum-entropy estimate [173] states that, given a set of estimates about a probability distribution, the one with the maximum information entropy contains the least bias. Since the QR ordering ranks sequences in order of decreasing diversity, taking the sequences from this ordering until the entropy of the profile is at a maximum (choosing the cutoff to maximize the information entropy) should result in the QR set with the least possible bias. Any additional sequences from the ordering will increase the bias of the profile, as measured by the information content of the individual positions.

Another way to approach the QR cutoff choice is by the definition of redundancy. Redundancy in information theory is defined as the difference in the maximum theoretical entropy rate of an information source (e.g., in an alignment, where each position has a uniform distribution of residues) and that actually observed:

\[
R = h_{\text{theor}} - h_{\text{obs}}.
\]  

If \( R > 0 \), then there is some degree of information redundancy. From Equation (4.3) it is apparent that to minimize redundancy, the entropy must be maximized. In a sequence alignment, this redundancy corresponds to information about the residues at a given position in an alignment. Considering bias to be an inflation of information about the positions in a profile, then one sees once again that a minimally biased QR ordering can be selected by choosing the cutoff to maximize the entropy.

In this study, we evaluate the effectiveness of the sequence QR method using a maximum-entropy cutoff in producing unbiased profiles from sequence alignments. We first generate simulated phylogenies using four evolutionary models producing increasing degrees of phylogenetic imbalance and then generate corresponding multiple sequence alignments. For each sequence alignment we determine a maximum-entropy evolutionary profile without using the tree of the simulated phylogeny. The bias in a profile is evaluated by comparing the imbalance of its phylogeny to that of the simulated phylogeny from which it was extracted.
Figure 4.5: Representative 100 taxon simulated phylogenies using four different values of speciation parameter $\sigma$.

and to the expected value for a balanced evolutionary model. We initially perform the analysis for protein sequences and then also for nucleic acid sequences, to determine if their reduced alphabet size has an impact on the independent site assumption. Finally, we compare the performance of the sequence QR method to the phylogenetic diversity method of selecting a representative set if a phylogenetic tree of the sequences is already known and show two examples of how evolutionary profiles can be used to improve sequence based analyses.

4.3.4 Methods

Phylogenetic simulations

To generate unbalanced phylogenies, the evolution of a clade of taxa was modeled using evolving speciation rates as described by Heard [174]. A log-Brownian model of rate evolution with punctuated change was used, meaning rates changed only at speciation events. In a simulation, a clade started as a single taxon with an initial speciation rate $\lambda$ and after each timestep (of duration $\Delta t$) all taxa in the clade randomly speciated with probability $\lambda \cdot \Delta t$. When a speciation event occurred, one taxon retained the old speciation rate and the other received a new rate according to the formula $\log(\lambda_{\text{new}}) = \log(\lambda_{\text{old}}) + \epsilon(\sigma)$, where $\epsilon(\sigma)$ was a normally distributed random value with expectation zero and standard deviation $\sigma$. $\Delta t$ was scaled during the simulation so the maximum speciation probability during any timestep was 0.01. The simulation ran until the clade had the desired number of taxa at which point the branching history from the simulation was saved as the true phylogenetic tree of the clade.

The parameter $\sigma$ defines how much speciation rates can change during speciation events. When $\sigma = 0$, the model reverts to the equal-rates Markov (ERM) model for “random” phylogenies commonly used as a model of null-imbalance (i.e., having a balanced tree) in the study of phylogenetic balance [175; 176]. Values of $\sigma > 0$ allow individual taxa to randomly acquire higher or lower speciation rates than the rest of clade and thus cause the final phylogeny to be unbalanced with their increased or lack of descendants. The effect of increasing $\sigma$ is to widen the distribution of speciation rates, permitting higher relative rates to
evolve and increasing the imbalance of the final phylogeny. Examples of typical simulated phylogenies for different values of $\sigma$ are illustrated in Figure 4.5.

**Tree shape measures**

To analyze the imbalance of our simulated phylogenies, four measures of tree shape were used that have been shown to have statistical power when testing trees for non-random branching [177; 178].

Colless’ Index [179], as corrected by Heard [176], is the sum over every internal node of the difference in the number of taxa descending along its right and left branches, normalized by the maximum imbalance for a tree with $n$ taxa:

$$I_C = \frac{2}{(n-1)(n-2)} \sum_{i=1}^{\text{nodes}} |S_i^R - S_i^L|.$$ 

$\bar{N}$ and $\sigma^2_N$ [180] are the mean and variance of the number of branchings between the root of the tree and every taxon:

$$\bar{N} = \frac{1}{n} \sum_{i=1}^{n} N_i,$$

$$\sigma^2_N = \frac{1}{n} \sum_{i=1}^{n} (N_i - \bar{N})^2.$$ 

$B_1$ [181] is the sum of the inverse of $M$ over every internal node except the root of the tree, where $M$ is the maximum number of branchings to any of the node’s descendent taxa:

$$B_1 = \sum_{i=1}^{\text{nodes}} \frac{1}{M_i}.$$ 

**Generated sequence alignments**

Generation of random sequences from simulated phylogenies was performed using Seq-Gen (version 1.3.2) [182]. Protein sequences were created using the BLOSUM62 [183] model for amino acid frequencies and substitution rates and nucleic acid sequences were created using the GTR model with nucleotide frequencies of $A=0.35$, $C=0.15$, $G=0.25$, $T=0.25$ and substitution rates of $A$–$C$=2.0, $A$–$G$=4.0, $A$–$T$=1.8, $C$–$G$=1.4, $C$–$T$=6.0, $G$–$T$=1.0 (parameters from [184]). Evolution rates were randomly assigned to the sites according to a continuous gamma distribution. After testing values of 0.5, 1.0, and 2.0 for the distribution’s shape parameter, a value of 2.0 was selected for the final analysis as producing the desired range of average percent identities (14–60% for protein sequences). The default program option of no invariable sites was also used, as these sites were not expected to have an impact on the analysis and excluding them allowed the use of shorter sequences.

**QR factorization of sequence alignments**

The details of applying the QR factorization to a sequence alignment to obtain an ordering of the sequences in terms of decreasing linear independence have been given previously [34] (including supporting informa-
Figure 4.6: Orthogonal encoding of a protein sequence alignment as an $L \times N \times D$ dimensional matrix.

tion), and will only be summarized here. The method begins by encoding the alignment as an $L \times N \times D$ dimensional matrix. Each sequence is a column in the $L \times N$ submatrix, so $L$ is the length of the sequence alignment and $N$ is the number of sequences it contains. The $D$ dimensions store an orthogonal encoding of each position in the sequence alignment. For protein sequences, $D$ has 24 components: one for each of the twenty amino acids, three for ambiguous residues (B, X, and Z), and one for a gap. For nucleic acid sequences $D$ has 5 components (A, C, G, T/U, and a gap). At a given position in the alignment, all components of $D$ are 0 except for the one corresponding to the residue type at that position, which contains a 1 if it is a physical residue or a scalable weight if it is a gap. Figure 4.6 illustrates an encoding of protein sequences. To perform the QR factorization of a sequence matrix, the sequence with the lowest average percent identity is pivoted to the first column and the following steps are performed successively for each column $k = 1, 2, ..., N$:

1) A Householder transformation is applied to each $L \times N$ submatrix independently in all $D$ dimensions to remove any linear contribution of column $k$ from columns $k + 1$ to $N$.

2) The $D$ dimensional Frobenius-like matrix $p$-norm ( $p$ equals 2 for proteins, 3 for nucleic acids) is calculated for columns $k + 1$ to $N$ to determine the next most independent sequence.

3) The column with the highest $p$-norm is pivoted to column $k + 1$ to prepare for the next round of factorization. The pivoted columns are tracked during the factorization and become the ordering of the sequences from most independent to least.

4.3.5 Statistical properties of maximum-entropy evolutionary profiles

To analyze the effect of the QR cutoff on the bias in EPs, we generated 10,000 phylogenies containing 100 taxa for each of four values of speciation parameter $\sigma$ (0.0, 0.1, 0.2, 0.3). These values of $\sigma$ resulted in phylogenies ranging from no to high imbalance. Protein sequences 100 residues in length were then generated for the taxa in each phylogeny to produce a gap-less multiple sequence alignment. Some closely related taxa were therefore over-represented in alignments from phylogenies with high imbalance. All possible EPs for each phylogeny were constructed from the alignment using the sequence QR method at every cutoff value. The resulting profiles were evaluated for entropy, percent identity, and phylogenetic imbalance as calculated by four measures of tree shape. The phylogenetic tree for an EP (needed for the
imbalance calculations) was created by simply removing the excluded taxa from the true phylogenetic tree of the simulation, no sequence based reconstruction was performed. Doing so eliminated the chance of inaccurate tree reconstructions due to long branch lengths obtained from the simulations. An example of the analysis on a representative phylogeny is shown in Figure 4.7.

Bias in a profile leads to asymmetry in the number of taxa found along different branches of its phylogenetic tree. This property allows the bias in a profile to be quantified in terms of the shape of its tree. If a tree has more imbalance than would be expected in a random tree of the same size, the profile is biased to some extent, whether by natural variation or man-made artifacts. To determine the level of bias in EPs from our simulations, the imbalance of their phylogenies was compared to that expected from phylogenies generated from the random equal-rates Markov (ERM) model. Since the expectation of each tree shape measure is not known analytically for ERM trees (except $I_C$ [185] and $\bar{N}$ [177]), we followed Kirkpatrick and Slatkin [177] and used simulations to estimate the distributions. Simulations of 10,000 ERM phylogenies of sizes from 10 to 100 taxa in increments of 5 were performed and from them the expectation and 95% confidence intervals of the imbalance measures were calculated. Our results agree with previous studies up to their maximum of 50 taxa and extend them out to 100 taxa. The distributions of the tree shape measures for unbalanced phylogenies were then estimated in the same way. These distributions permit a determination as to whether an EP’s phylogenetic tree corresponds to a balanced or an unbalanced phylogeny and, consequently, if the profile is biased.

Comparing the imbalance of phylogenetic trees of EPs calculated from unbalanced phylogenies to the ERM expectation (Figure 4.8) one can see that at a cutoff of 90% identity or higher the EPs’ phylogenetic
trees fall outside of the 95% confidence interval for being balanced. Their profiles, while best representing the phylogeny at the cutoff, still contain a great deal of the over-representation initially present in the full alignment. As the cutoff is lowered, excluding more and more closely related sequences, the mean imbalance approaches the expectation of ERM phylogenies until the curves converge between 30–40% identity. This same pattern holds regardless of the imbalance of the initial phylogeny (data not shown). At 30% identity, however, the EPs on average contain only 8 of the original 100 sequences. Since this level of identity is known to be near the “twilight zone” where homology for proteins becomes questionable using sequence methods [186], a reasonable assumption would be that the additional diversity gained by including some of the excluded sequences in the profile would be worth the trade-off.

As discussed earlier, information entropy can be used as an impartial judge as to whether the increase in redundancy from adding additional sequences is compensated by the additional diversity they provide. To test this, the information entropy of every EP previously created from each phylogeny was calculated, resulting in the entropy of the EP as a function of profile size. In every case, this function was monotonically increasing up to a single maximum and monotonically decreasing afterward. The profile at the maximum entropy value was taken to be the phylogeny’s maximum-entropy evolutionary profile (ME-EP). Figure 4.9 shows the results of analyzing the imbalance of the phylogenetic trees of the ME-EPs grouped by $\sigma$. For balanced phylogenies ($\sigma=0.0$), the mean imbalance follows the expectation of the model used to generate the phylogenies. This agrees with our conjecture that, since ERM phylogenies contain no imbalance, their alignments contain no over-representation that the ME-EPs can exclude. The generated phylogeny and the ME-EP’s phylogenetic tree both have identical, and minimal, amounts of imbalance. For unbalanced phylogenies ($\sigma>0.0$), though, the mean imbalance of the ME-EPs’ phylogenetic trees and that expected from the models used to generate the phylogenies do not follow the same curve. The imbalance of the ME-EPs’ phylogenetic trees is significantly lower. In fact, the imbalance of phylogenetic trees of ME-EPs from unbalanced phylogenies closely follows that expected for ERM phylogenies. Since ERM phylogenies are
Figure 4.9: Imbalance statistics for phylogenetic trees of maximum-entropy evolutionary profiles of 10,000 simulated phylogenies using speciation parameter $\sigma$ from 0.0 (top row) to 0.3 (bottom row). Solid lines give the expectation and 95% confidence intervals for phylogenies generated with the specified value of $\sigma$, while dotted lines give the expectation and 95% confidence intervals for balanced phylogenies ($\sigma=0.0$; ERM model). Gray points show the imbalance values of the phylogenetic tree of the maximum-entropy evolutionary profile from a random sample of the simulated phylogenies and red points mark the mean imbalance for a given number of taxa.

In the model of null imbalance, these results show that ME-EPs contain a nearly unbiased subset of an initial alignment where some taxa are over-represented.

Phylogenetic trees of ME-EPs have, on average, the same imbalance as profiles constructed using a cutoff of 60% identity, but the actual maximum percent identity in an ME-EP varies, with the distribution ranging from 40–100% as shown in Figure 4.10. Choosing an identity cutoff for the sequence QR method of less than 40% will therefore always generate a profile that is a subset of the ME-EP. These profiles will, of course, have phylogenetic trees with imbalance less than or equal to those of the ME-EP. However, they achieve their non-redundancy by sacrificing diversity. There are sequences that could be added to them whose diversity would compensate for any redundancy. This is precisely the trade-off that the maximum-entropy method makes, in a systematic way. Compared to profiles generated with a cutoff of 60%, ME-
EPs contain more sequences but have the same average amount of imbalance in their phylogenetic trees because the maximum percent identity is adjusted automatically on a profile-by-profile basis. One can also see from Figure 4.10 that the mean maximum identity in the ME-EPs slowly increases as does the imbalance of the simulated phylogenies. This is because at high levels of imbalance many sequences are very closely related. If any sequence diversity at all is to be captured, it means including some closely related sequences. For example, in a set with a minimum percent identity of 80% one likely would not be satisfied with a profile containing only a single sequence, which is what would be obtained from a typical cutoff. A few sequences would likely be desired to get a sense of the possible variations. The maximum-entropy method automatically accounts for this by maximizing the diversity while minimizing information theoretic redundancy in the profile.

### 4.3.6 Extension to nucleic acid sequences

Considering the importance of RNA and DNA, a logical extension to the sequence QR method is to allow for the creation of profiles of nucleic acid sequences. The difference between proteins and nucleic acids from a sequence perspective is the size of the alphabet; nucleic acids are composed of only four nucleotides (A, C, G, T/U) as opposed to the 20 amino acids of proteins. The QR factorization algorithm was modified in a straightforward way to use the smaller alphabet and the appropriate norm and gap-scaling parameters (3 and 0.5 respectively) were determined in the same manner as for proteins in the original work [34]. The appropriate method for using information entropy to determine the QR cutoff for nucleic acids, however,
was less obvious. Two random protein sequences will have the same residue at the same position 1/20th of the time, but in nucleic acids the chance increases to 1 in 4, i.e., conservation of any single site is much less significant in nucleic acids. This suggested that it might be more difficult to determine, using information entropy, if an additional sequence is redundant or not since the total entropy value for an alignment is the sum of the scores for each position independently.

Sequences of biological origin are not simply a series of independent positions. They commonly do have correlations between sites due to functional constraints. To account for correlations within a sequence when calculating entropy, previous authors have used Shannon’s theorem of source entropy to estimate it using blocks of symbols [187; 188]. For a sequence broken into symbol blocks of length $B$, the entropy per block is given by

$$H(B) = - \sum_{i_1=1}^{n} \cdots \sum_{i_B=1}^{n} p(i_1, \cdots, i_B) \log_2 p(i_1, \cdots, i_B),$$

where $p(i_1, \cdots, i_B)$ is the probability of the 1st symbol being in state $i_1$ and the 2nd symbol in state $i_2$ and so forth. Shannon’s theorem states that the source entropy per symbol of a sequence after accounting for any short or long range correlations is given by

$$h = \lim_{B \to \infty} \frac{H(B)}{B}.$$ 

This method relies on $B$ being large and estimating the probabilities of large blocks using symbol frequencies leads to an underestimation of the entropy due to finite sampling. Various ways to correct the estimate have been proposed [189], but since the study presented here only compares entropy values for sets with the same $B$, these corrections are ignored.

The effect of block size during the creation of nucleic acid ME-EPs was studied by comparing ME-EPs generated by maximizing the independent entropy ($B=1$) to those generated by maximizing several block entropy estimates (block sizes from 2–5). Figure 4.11 shows the results of these studies. Compared to the values seen for protein ME-EPs, the imbalance of nucleic acid profiles using block sizes of one or two look reasonably unbiased. The higher block entropy estimates ($B > 2$), on the other hand, show increased imbalance compared to protein ME-EPs. The mean size of the profiles created using a block size of two is somewhat larger than for profiles created assuming independent sites. This difference may indicate that
a block size of one under-samples the phylogeny. Additional studies using data containing correlations between sites are required to further quantify this effect. The difference is small enough, though, that it would appear that the independent site assumption works reasonably well in regard to choosing an ME-EP for nucleic acids.

4.3.7 Interpreting the size of a profile

The number of sequences required for an ME-EP to represent the sequence diversity manifested by a biological sequence may provide information about the evolutionary constraints on the sequence. By analogy with Shannon’s definition of sequences as messages from a data source, one can consider each sequence in an alignment as one out of the full set of possible “equivalent” sequences. Here, equivalent means those sequences produced by the evolutionary process that, in a broad sense, perform the same function. If the full range of equivalent sequences was known, it would define the portion of the full sequence space that was compatible with the biological function. Shannon showed that the number of sequences that are likely to be produced by a source is $2^{H_{\text{seq}}}$ (assuming $H_{\text{seq}}$ is given in bits, i.e., the entropy logarithm is base 2), so the problem of estimating the size of the subset of the full sequence space spanned by a biological function can be thought of in terms of estimating the entropy of the sequences that perform the function. Unfortunately, due to finite sampling and other effects, estimating the absolute value of the sequence entropy is difficult. For the purposes of this work it is simply noted that for a multiple sequence alignment containing all known sequences of a given function, the subset with the highest entropy best spans the full space.

The number of sequences in an ME-EP can therefore be used to compare the sizes of the sequence space of different biological molecules. If alignments are available for two molecules, each containing sequences from the same species, then the one with the smaller ME-EP has a reduced sequence space. For example, elongation factor Tu (EF-Tu) and ribosomal protein S4 are similarly sized proteins with universal distribution among the domains of life. The ME-EP sizes for the two molecules are 22 and 49, respectively. The difference between the two indicates that EF-Tu is evolving in a more compact sequence space than is S4. This is in accordance with a well-known property of EF-Tu, that it must bind to several different partners and thus is evolutionarily constrained. A similar example can be shown with the 23S ribosomal RNA. The 23S molecule is often divided into 6 subdomains of similar size for analysis. If the ME-EP of each subdomain is generated from an initial alignment of 475 sequences, the sizes of the profiles are, in order: 29, 21, 50, 15, 13, and 25. One can quickly see that domains 4 and 5 have the smallest profiles. A sequence conservation analysis of the 23S reveals a similar trend; domains 4 and 5 are more conserved than the others.

A statistical theory of combinatorial protein libraries, which attempts to estimate the number of sequences that satisfy a particular protein fold, is under development by others [190]. In this theory, the probabilities of different residues occurring at a specific position are determined by maximizing the sequence entropy subject to energetic constraints taken from the native structure. It would be interesting to compare the sequence space size predicted by theory against the size actually explored by the evolutionary process. Such a study might lead to a deeper understanding of how evolution explores a space.
4.3.8 Relationship to phylogenetic diversity

If the phylogenetic tree of a sequence alignment is available, EPs generated using the sequence QR method are, at least conceptually, similar to a technique introduced by Faith [191]. Faith outlined a method to select the most diverse subtree of a given size from a phylogenetic tree using the phylogenetic diversity (PD) measure, which is defined as the sum of the branch lengths connecting all of the taxa in a tree. The procedure is simply to find the one subtree, out of all possible subtrees of the desired size, that has the maximum PD. Since its introduction, PD has been proposed for and used in prioritizing and measuring the progress of biodiversity conservation efforts [192–196]. Recently, methods have been implemented for efficiently selecting the subtree with the maximal PD from a phylogenetic tree using the greedy algorithm [197; 198] and for determining the best selection subject to constraints [199; 200]. It has also been proposed that the maximal PD method could be used to select future genomes for sequencing [201].

Figure 4.12 shows the percentage of PD captured as a function of the set size for both the QR and maximum PD methods. The plots are averages over 10,000 simulated phylogenies at four different levels of speciation parameter $\sigma$. The maximal PD sets were obtained by running the PDA program [198] in the gPDA mode on the true phylogenetic tree from the simulation. From the figure it is apparent that the QR method selects sets that are practically equivalent to the maximal PD sets, at high levels of species bias. With no bias, the QR sets are slightly less representative in terms of PD than the maximal PD sets, but even this difference is at most 2.5% of the total. It is also worthwhile to note that the QR method analyzes only the sequence data.
and never the true phylogenetic tree. When the maximal PD method is used on phylogenetic trees generated using Clustal W [202] from the simulated sequences (a more equivalent situation) the difference narrows to at most 1.7% of the total PD. The small average difference between the PD of a QR set and the maximum possible PD indicates that the QR ordering closely follows the topology of the true phylogenetic tree.

The fundamental difference between the PD and QR methods is that PD selects diversity according to the criteria used to construct the tree while QR operates directly on the sequence data. Given a phylogenetic tree including branch lengths from which one wishes to select a representative set, the maximal PD method will provide a quicker answer. In contrast, if one is working with sequence data the QR method provides a more direct approach to obtaining a reduced but representative set before undertaking the process of generating a phylogenetic tree. The two methods are alike, however, in their lack of a way to measure the redundancy included in a given set. Neither method by itself can make a determination as to how many taxa should be included in a profile before it becomes biased. The maximum-entropy cutoff developed in this study provides this capability for the QR method and could also be used to determine a cutoff for a maximal PD selection if the tree was generated using molecular sequence data. In fact, since the PD method starts with a phylogenetic tree, a model based estimation of the residue probabilities for the entropy calculation could be obtained, as opposed to the non-model based estimate used in the QR construction of ME-EPs without a phylogenetic tree.

4.3.9 Covariation detection with mutual information

As a final example of the utility of using ME-EPs, consider the problem of using mutual information to detect covariation. Mutual information (MI) is the reduction in entropy of one random variable \(X\) given knowledge of the state of another \(Y\):

\[
MI(X,Y) = H(X) - H(X|Y),
\]

\[
= H(X) + H(Y) - H(X,Y),
\]

where \(H(X,Y)\) is the joint entropy of the two variables. If the variables are independent, no knowledge is gained and the MI is zero. If they are completely dependent, complete knowledge is gained and the MI is the minimum of \(H(X)\) or \(H(Y)\). In a biological sequence alignment, if a mutation in one position correlates with a compensating change in another, the two residues will have a high MI value. A sequence alignment can therefore be analyzed for covarying residues, although the technique suffers from signal problems due to normal phylogenetic covariation [203]. MI has previously been used to detect structural contacts, clusters of interrelated residues, and protein-protein interactions [39; 204]. Here we apply it to detecting coevolution in protein-RNA interfaces.

To ensure that an MI analysis produces the best possible results, the sequence profile should be as unbiased as possible. Any bias in the profile will artificially increase the MI by lowering the joint entropy values due to the over-representation of residue pairs associated with the biased sequences. The ME-EP minimizes such bias and should therefore perform better than other profiles. To test this, we attempted to
Figure 4.13: Maps of the interface between *Thermus thermophilus* ribosomal protein S4 and 16S ribosomal RNA using contact distance (left), mutual information based on a maximum-entropy evolutionary profile (center), and mutual information based on a biased profile (right). The contact distance ranges from 0 Å (black) to 20+ Å (white) and the normalized mutual information from 0 (white) to 1 (black). Only mutual information values greater than 2 standard deviations above the mean are shown.

Detect coevolving residues at the protein-RNA interface of ribosomal protein S4 and the 16S ribosomal RNA (rRNA) of *Thermus thermophilus*. Alignments of S4 and 16S containing bacterial and archaeal sequences were created from the public databases. Both an ME-EP and a random bacterially-biased profile of the same size were constructed from these alignments and used to calculate the pairwise MI values between *T. thermophilus* S4 and 16S, computed using Equation (4.4) and normalized by joint entropy [39]. Figure 4.13 shows the contact map (based on the atomic structure 2J00 of Selmer et al. [15]) and MI maps for S4 and the 16S. One can see that there are numerous residue pairs with high MI, between both contacts and non-contacts. Filtering the values using a distance constraint limits the results to those residues that are interacting at the interface. The results using the ME-EP reveal that nucleotides 401–431 (corresponding to rRNA helix H16) have high MI with the protein. A detailed analysis has shown that these nucleotides are a bacterial specific rRNA insertion that is coevolving with an insertion (residues 21–37) in the bacterial version of ribosomal protein S4. In contrast, the biased profile does not show the same region as significant due to the over-representation of bacteria.

While the above example of two co-evolving insertions presents an obvious case, ME-EPs can also increase the signal when searching for more subtle MI features, including the identification of phylogenetic signatures in well-aligned regions [46].

### 4.3.10 Conclusions

Over-representation is an ongoing source of difficulty when analyzing biological sequence data. Creating a unbiased set of sequences that are representative of the sequence space explored by evolution as the first step in an analysis can increase the power of bioinformatics techniques. The sequence QR method using the maximum-entropy cutoff creates just such a representative set with a minimal amount of bias from
an initial arbitrarily biased sample. These maximum-entropy evolutionary profiles can be created from both protein and nucleic acid sequence alignments and have proven useful as part of a mutual information analysis to detect coevolution in protein-RNA interfaces. ME-EPs can also be informative irrespective of any further analysis performed with them. The size of the profile gives an indication of the diversity present in the sequences it represents. A small profile that spans a large amount of biological diversity may be an indication of strong constraints acting to limit the amount of sequence space compatible with the function. Conversely, a large profile implies a greater amount of evolutionary flexibility in the sequences.
Chapter 5

Lattice Microbe Method for Modeling Cellular Reaction Networks

5.1 Background

The cell is a crowded space [205; 206] with proteins, nucleic acids, and other macromolecules constantly in contact with and colliding into each other. In the midst of this chaotic and turbulent scene, extensive and intricate networks of biochemical reactions [207; 208] operate, in many cases, by random Brownian diffusion of one molecular species to its reaction counterpart. Often the concentration of one or both of the reactants is as low as a few molecules per cell, resulting in stochastic dynamics that depend upon a molecule’s initial position. Additionally, the cell is not a homogeneous mixture of macromolecules, some are localized to specific sub-volumes within the cell and their localization has a dramatic effect on biochemical networks in which they participate [209; 210]. Computational models of cellular biochemical networks that are spatially resolved can therefore be useful when testing hypotheses developed from single molecule experimental results of a network’s activity, looking for unexpected, emergent behavior in a network’s temporal and spatial dynamics, and comparatively investigating the parameter space explored during a network’s evolution.

While spatially accurate models of biomolecular networks have been ascribed with the potential of enabling new studies of cellular biochemical systems [211; 212], developing them is a challenging endeavor. First, the spatial organization of the cell must be known [213]. The many varieties of fluorescence microscopy [214–216] have provided a recent explosion in the availability of data regarding in vivo spatial positioning (20–30 nm resolution in some cases). Another technique that can yield data on the in vivo positioning of macromolecules is cryoelectron tomography, which has been used to localize ribosomes in intact cells with ~5 nm resolution [217]. It is anticipated that by combining results from these types of studies the global distribution of large macromolecules within the cell can be reasonably approximated for various parts of the cell cycle.

Another key issue in modeling whole-cell biochemical networks is accounting for the cellular environment. Inside a cell, approximately 20–30% of the volume is occupied by macromolecules, the diffusion coefficients of which are reduced 3 to 15 fold relative to their in vitro values [218; 219]. Additionally, diffu-
sive behavior that does not obey the standard relation between mean square displacement (MSD) and time has been observed in living cells. Specifically, anomalous subdiffusion has been seen in both experimental [220–222] and theoretical studies [223–225], although its exact extent (and origin) inside living cells is still debated [219; 226; 227]. It is also clear that macromolecular crowding has an effect on reaction kinetics [228]. The enhanced concentration effect due to particle localization under in vivo conditions can produce large total changes in the dynamics of a reaction.

Once a spatial model of the cell has been constructed, there are computational challenges associated with simulating the model, both in terms of calculating the dynamics of the individual macromolecules and of calculating their interactions with the in vivo environment (for two recent reviews, see [229] and [230]). Thus far, promising approaches toward addressing the problem have involved either solving the reaction-diffusion master equation for sub-volumes [231] or using Brownian dynamics methods to simulate random movements of individual macromolecules [225; 232; 233]. However, neither of these methods are currently able to simulate a whole cell under in vivo conditions. Reaction-diffusion master equation methods efficiently capture cell-scale spatial and temporal dynamics, but have not been able to account for in vivo environments. Brownian dynamics methods can simulate in vivo crowding, but must calculate pair interactions limiting their ability to simulate over cellular length and time scales for crowded systems (millions of molecules).

We have derived a new cellular automata (CA) [234–236] based method that utilizes the graphics processing unit (GPU) to perform long time-scale simulations of whole-cell reaction-diffusion models under in vivo conditions. CA methods have long been used in statistical physics and computational chemistry [237–240] and ours is a derivative of a multiparticle model [241]. Being a lattice model, computational complexity scales with the number of lattice sites (independent of the number of particles located on the lattice) and so can reach long time-scales under crowded conditions.

The emergence of the GPU as a widely-available, dedicated compute processor has recently had a profound impact on scientific computing [242; 243]. This development is due, in part, to the release by NVIDIA of the Compute Unified Device Architecture (CUDA) API [244] allowing general-purpose programming of the GPU in a C-like language. The GPU provides a massively parallel architecture, capable of performing hundreds of simultaneous calculations, but each calculation is highly constrained in terms of its ability to access global resources. It performs best under conditions of high calculation/data density and localized memory use, where calculations are independent of each other.

CA models are characterized by three properties: space and time are discrete, physical quantities are described by a finite set of values, and the time evolution of the system is governed by a rule using only local information. These properties make them theoretically well-suited toward GPU implementation, since calculations use integer math (no dependency on GPU floating point precision) and the entire lattice can be updated in parallel using only small amounts of local memory. Additionally, CA models are highly parallelizable, most of the code can run on the GPU. CA methods have previously been implemented on parallel architectures [245; 246] and while the insights gained are useful for a GPU implementation, the techniques themselves are not directly transferable.

In this work we present the techniques and strategies that enabled simulations of in vivo reaction-
diffusion on GPU hardware. We first introduce the multiparticle model and the adaptations made to it to permit efficient GPU simulation. We then discuss the implementation details of the method and the programmatic trade-offs made to accommodate GPU architectural limitations. Finally, we present analyses of free and obstructed diffusion simulations along with timing results from whole-cell in vivo simulations.

5.2 Diffusion Processes

5.2.1 Multiparticle diffusion model

Multiparticle diffusion models are a class of phenomenological diffusion model; they are lattice-based models in which particles follow independent random walks between lattice sites in a stochastic manner. Multiparticle models are characterized by allowing multiple particle per lattice site, as opposed to lattice-gas-automata based diffusion models in which a particle completely occupies a site, excluding other particles from moving to it.

We introduce a multiparticle model based on that of Karapiperis and Blankleider [241] with modifications to support efficient implementation on a GPU. The model is constructed on a cubic lattice \( \mathcal{L} \) with uniform spacing in the x-, y-, and z-dimensions with distance \( \lambda \). Lattice sites are located on the lattice at positions \( \mathbf{r} = a\lambda \mathbf{i} + b\lambda \mathbf{j} + c\lambda \mathbf{k} \), where \( a, b, \) and \( c \) are integers. Particles of various species \( \alpha \) are positioned at the lattice sites according to some initial condition at time \( t = 0 \) and then move on the lattice from site to site according to the rules of the model. Time is also discrete in the model and particle movement occurs instantaneously at time steps separated by time \( \tau \).

At time \( t \), a site at position \( \mathbf{r} \) on the lattice is described by its occupancy, \( N_\alpha(\mathbf{r}, t) \), giving the number of particles of species \( \alpha \) located at the site. A diffusion operator \( \mathcal{D} \) updates a lattice site at a time step by moving particles to and from the site according to a model of their diffusive behavior. The state of a lattice site after a single time step is therefore given by

\[
N_\alpha(\mathbf{r}, t + \tau) = \mathcal{D} N_\alpha(\mathbf{r}, t).
\]  

The time evolution of the entire lattice is realized by the simultaneous application of the diffusion operator on each lattice site.

Multiparticle diffusion microdynamics

Normal Brownian diffusion can be phenomenologically modeled as a series of independent random choices for the movement of particles in a system. In previous models [236; 241], particles could move only in a single dimension during a time step, i.e., a particle moved \( \pm \lambda \mathbf{i}, \pm \lambda \mathbf{j}, \) or \( \pm \lambda \mathbf{k} \). However, implementing such a model requires access to the entire three-dimensional neighborhood surrounding each lattice site during a time step. As will be shown later (Section 5.2.2), this requirement severely limits the computational performance of the model on a GPU. We instead model diffusion as three independent random choices (one for each dimension) during a single time step. Decomposing the problem in such a manner reduces the size
of the neighborhood that must be accessed to perform the diffusion calculation. During each fractional time step only the neighborhood in the dimension being process is required and GPU performance is dramatically improved.

At each time step in our multiparticle diffusion model, for each dimension, a particle has a probability of moving one lattice site in the negative direction, a probability of staying at the current lattice site, and a probability of moving one lattice site in the positive direction. For the x, y, and z dimensions, these probabilities are \((p_{-1}, p_0, p_1)\), \((q_{-1}, q_0, q_1)\), and \((s_{-1}, s_0, s_1)\), respectively. Since a particle must make a single choice in each dimension, \(p_{-1} + p_0 + p_1 = q_{-1} + q_0 + q_1 = s_{-1} + s_0 + s_1 = 1\). Independent diffusion in each of the three principle dimensions implies that the neighborhood a particle can move to after a time step is composed of all twenty-six nearest neighbors. The probability of moving to any one is the product of the p, q, and r probabilities required to make the specific x-y-z move. Figure 5.1 shows the probabilities for a particle to move from a site \(\vec{r}\) to any of the neighboring sites or to remain at \(\vec{r}\) during a time step.

**Diffusion operator**

The diffusion operator applied to a lattice site calculates the random movement choices for each nearby particle and then updates the occupancy of the site to be the sum of all particles that remain at the site and those that enter from neighboring sites,

\[
D N_\alpha(\vec{r}, t) = \sum_{a=-1}^{1} \sum_{b=-1}^{1} \sum_{c=-1}^{1} \sum_{n=1}^{1} N_\alpha(\vec{r} + \vec{d}_n, t) \theta(n, \vec{r} + \vec{d}_n - \vec{d}_n, t),
\]

where \(\vec{d}_n = a\lambda\hat{i} + b\lambda\hat{j} + c\lambda\hat{k}\). The function \(\theta(n, \vec{r}, \vec{d}_n, t)\) is a stochastic function returning 1 if particle \(n\) at site \(\vec{r}\) moves \(\vec{d}_n\) at time \(t\), otherwise 0. It functions according to the probabilities previously defined for the moves, such that

\[
P(\theta(n, \vec{r}, a\lambda\hat{i} + b\lambda\hat{j} + c\lambda\hat{k}, t) = 1) = p_a q_b s_c.
\]
Additionally, since each particle must make one and only one choice during each timestep, it follows that
\[
\sum_{a=-1}^{1} \sum_{b=-1}^{1} \sum_{c=-1}^{1} \theta(n, \vec{r}, a\lambda\hat{i} + b\lambda\hat{j} + c\lambda\hat{k}, t) = 1.
\]

In a statistical ensemble, then,
\[
N_\alpha(\vec{r}, t) < \sum_{n=1}^{N_\alpha(\vec{r}, t)} \theta(n, \vec{r}, a\lambda\hat{i} + b\lambda\hat{j} + c\lambda\hat{k}, t) >= p_a q_b s_c N_\alpha(\vec{r}, t),
\]
and, in the macroscopic limit,
\[
\mathcal{D} N_\alpha(\vec{r}, t) = \sum_{a=-1}^{1} \sum_{b=-1}^{1} \sum_{c=-1}^{1} p_{-a} q_{-b} s_{-c} N_\alpha(\vec{r} + d\vec{r}, t). \tag{5.2}
\]

**Lattice time evolution**

The time evolution of the entire lattice is realized by the simultaneous application of the diffusion operator on each lattice site. It can be examined to analyze the diffusive behavior of the model, following Chopard and Droz [236]. Since there are no diffusive interactions between particles of different species, each species diffuses independently on the lattice and we drop the \( \alpha \) qualifier from the site occupancy expression for simplicity. From Equations (5.1) and (5.2), the occupancy of a lattice site after a time step is given by
\[
N(\vec{r}, t + \tau) = \sum_{a=-1}^{1} \sum_{b=-1}^{1} \sum_{c=-1}^{1} p_{-a} q_{-b} s_{-c} N(\vec{r} + d\vec{r}, t).
\]

Writing out each term of the summation, we obtain the full expression for the time evolution,
\[
N(\vec{r}, t + \tau) = p_{1q1s1} N(\vec{r} - \hat{i} - \lambda\hat{j} - \lambda\hat{k}, t) + p_{0q1s1} N(\vec{r} - \lambda\hat{j} - \lambda\hat{k}, t) + p_{-1q1s1} N(\vec{r} + \hat{i} - \lambda\hat{j} - \lambda\hat{k}, t) + p_{1q0s1} N(\vec{r} - \lambda\hat{i} - \lambda\hat{j} - \lambda\hat{k}, t) + p_{0q0s1} N(\vec{r} - \hat{i} - \lambda\hat{j} - \lambda\hat{k}, t) + p_{-1q0s1} N(\vec{r} + \lambda\hat{i} - \lambda\hat{j} - \lambda\hat{k}, t) + p_{1q1s0} N(\vec{r} - \lambda\hat{i} - \hat{j}, t) + p_{0q1s0} N(\vec{r} - \hat{j}, t) + p_{-1q1s0} N(\vec{r} + \lambda\hat{i} - \hat{j}, t) + p_{1q0s0} N(\vec{r} - \hat{i}, t) + p_{0q0s0} N(\vec{r}, t) + p_{-1q0s0} N(\vec{r} + \lambda\hat{i}, t) + p_{1q1s-1} N(\vec{r} - \hat{i} + \lambda\hat{j} + \lambda\hat{k}, t) + p_{0q1s-1} N(\vec{r} - \lambda\hat{j} + \lambda\hat{k}, t) + p_{-1q1s-1} N(\vec{r} + \hat{i} - \lambda\hat{j} + \lambda\hat{k}, t) + p_{1q0s-1} N(\vec{r} - \hat{i} + \lambda\hat{k}, t) + p_{0q0s-1} N(\vec{r} + \lambda\hat{j} + \lambda\hat{k}, t) + p_{-1q0s-1} N(\vec{r} + \hat{i} + \lambda\hat{j} + \lambda\hat{k}, t).
\]

Expanding both sides around \( N(\vec{r}, t) \) and collecting terms gives the change in occupancy with respect to
time,
\[
\tau \frac{\partial}{\partial t} N(\vec{r}, t) + \frac{\tau^2}{2} \frac{\partial^2}{\partial t^2} N(\vec{r}, t) = 
\]
\[
- N(\vec{r}, t) + \left( \sum_{a=-1}^{1} \sum_{b=-1}^{1} \sum_{c=-1}^{1} p_a q_b s_c \right) N(\vec{r}, t) 
\]
\[
+ \lambda \frac{\partial}{\partial x} N(\vec{r}, t) \left( p_1 q_1 s_1 - p_{-1} q_{-1} s_{-1} - p_{-1} q_{-1} s_1 + p_1 q_1 s_{-1} \right) 
\]
\[
- p_1 q_1 s_0 + p_{-1} q_{-1} s_0 - p_{-1} q_0 s_0 + p_1 q_1 s_0 + p_{-1} q_{-1} s_0 
\]
\[
- p_1 q_{-1} s_{-1} + p_{-1} q_1 s_{-1} - p_{-1} q_1 s_{-1} + p_{-1} q_{-1} s_{-1} \right) 
\]
\[
+ \lambda \frac{\partial}{\partial y} N(\vec{r}, t) \left( p_1 q_1 s_1 + p_0 q_1 s_1 - p_{-1} q_1 s_1 + p_{-1} q_{-1} s_1 + p_0 q_{-1} s_1 + p_{-1} q_{-1} s_1 
\]
\[
+ p_1 q_{-1} s_0 + p_0 q_1 s_0 + p_{-1} q_{-1} s_0 + p_{-1} q_{-1} s_{-1} + p_0 q_{-1} s_{-1} + p_{-1} q_{-1} s_{-1} 
\]
\[
+ p_1 q_{-1} s_{-1} \right) 
\]
\[
+ \lambda \frac{\partial}{\partial z} N(\vec{r}, t) \left( p_1 q_1 s_1 + p_0 q_1 s_1 + p_{-1} q_1 s_1 + p_{-1} q_{-1} s_1 + p_0 q_{-1} s_1 + p_{-1} q_{-1} s_1 
\]
\[
+ p_1 q_{-1} s_0 + p_{-1} q_{-1} s_0 + p_{-1} q_{-1} s_{-1} + p_0 q_{-1} s_{-1} + p_{-1} q_{-1} s_{-1} 
\]
\[
+ \lambda \frac{\partial^2}{\partial x^2} N(\vec{r}, t) \left( p_1 q_1 s_1 + p_0 q_1 s_1 + p_{-1} q_1 s_1 + p_{-1} q_{-1} s_1 + p_0 q_{-1} s_1 + p_{-1} q_{-1} s_1 
\]
\[
+ p_1 q_{-1} s_0 + p_{-1} q_{-1} s_0 + p_{-1} q_{-1} s_{-1} + p_0 q_{-1} s_{-1} + p_{-1} q_{-1} s_{-1} 
\]
\[
+ \lambda \frac{\partial^2}{\partial y^2} N(\vec{r}, t) \left( p_1 q_1 s_1 + p_0 q_1 s_1 + p_{-1} q_1 s_1 + p_{-1} q_{-1} s_1 + p_0 q_{-1} s_1 + p_{-1} q_{-1} s_1 
\]
\[
+ p_1 q_{-1} s_0 + p_{-1} q_{-1} s_0 + p_{-1} q_{-1} s_{-1} + p_0 q_{-1} s_{-1} + p_{-1} q_{-1} s_{-1} 
\]
\[
+ \lambda \frac{\partial^2}{\partial x \partial y} N(\vec{r}, t) \left( p_1 q_1 s_1 + p_0 q_1 s_1 + p_{-1} q_1 s_1 + p_{-1} q_{-1} s_1 + p_0 q_{-1} s_1 + p_{-1} q_{-1} s_1 
\]
\[
+ p_1 q_{-1} s_0 + p_{-1} q_{-1} s_0 + p_{-1} q_{-1} s_{-1} + p_0 q_{-1} s_{-1} + p_{-1} q_{-1} s_{-1} 
\]
\[
+ \lambda \frac{\partial^2}{\partial x \partial z} N(\vec{r}, t) \left( p_1 q_1 s_1 + p_0 q_1 s_1 + p_{-1} q_1 s_1 + p_{-1} q_{-1} s_1 + p_0 q_{-1} s_1 + p_{-1} q_{-1} s_1 
\]
\[
+ p_1 q_{-1} s_0 + p_{-1} q_{-1} s_0 + p_{-1} q_{-1} s_{-1} + p_0 q_{-1} s_{-1} + p_{-1} q_{-1} s_{-1} 
\]
\[
+ \lambda \frac{\partial^2}{\partial y \partial z} N(\vec{r}, t) \left( p_1 q_1 s_1 + p_0 q_1 s_1 + p_{-1} q_1 s_1 + p_{-1} q_{-1} s_1 + p_0 q_{-1} s_1 + p_{-1} q_{-1} s_1 
\]
\[
+ p_1 q_{-1} s_0 + p_{-1} q_{-1} s_0 + p_{-1} q_{-1} s_{-1} + p_0 q_{-1} s_{-1} + p_{-1} q_{-1} s_{-1} \right),
\]
If the probability of remaining at a lattice site is isotropic (in each dimension must also equal one),
we can simplify the above to,
\[
\begin{align*}
\frac{\tau}{\partial t}N(\vec{r}, t) + \frac{\tau^2}{2} \frac{\partial^2}{\partial \xi^2} N(\vec{r}, t) &= -N(\vec{r}, t) + \left( \sum_{a=1}^{1} \sum_{b=1}^{1} \sum_{c=1}^{1} p_a q_b s_c \right) N(\vec{r}, t) \\
+ \lambda \frac{\partial}{\partial x} N(\vec{r}, t) (p_{-1} - p_1) (q_1 + q_0 + q_{-1}) (r_1 + r_0 + r_{-1}) \\
+ \lambda \frac{\partial}{\partial y} N(\vec{r}, t) (q_{-1} - q_1) (p_1 + p_0 + p_{-1}) (r_1 + r_0 + r_{-1}) \\
+ \lambda \frac{\partial}{\partial z} N(\vec{r}, t) (r_{-1} - r_1) (p_1 + p_0 + p_{-1}) (q_1 + q_0 + q_{-1}) \\
+ \frac{\lambda^2}{2} \frac{\partial^2}{\partial x^2} N(\vec{r}, t) (p_{-1} + p_1) (q_1 + q_0 + q_{-1}) (r_1 + r_0 + r_{-1}) \\
+ \frac{\lambda^2}{2} \frac{\partial^2}{\partial y^2} N(\vec{r}, t) (q_{-1} + q_1) (p_1 + p_0 + p_{-1}) (r_1 + r_0 + r_{-1}) \\
+ \frac{\lambda^2}{2} \frac{\partial^2}{\partial z^2} N(\vec{r}, t) (r_{-1} + r_1) (p_1 + p_0 + p_{-1}) (q_1 + q_0 + q_{-1}) \\
+ \lambda^2 \frac{\partial^2}{\partial x \partial y} N(\vec{r}, t) ((p_1 q_1 + p_{-1} q_{-1}) - (p_1 q_{-1} + p_{-1} q_1)) (r_1 + r_0 + r_{-1}) \\
+ \lambda^2 \frac{\partial^2}{\partial x \partial z} N(\vec{r}, t) ((p_1 r_1 + p_{-1} r_{-1}) - (p_1 r_{-1} + p_{-1} r_1)) (q_1 + q_0 + q_{-1}) \\
+ \lambda^2 \frac{\partial^2}{\partial y \partial z} N(\vec{r}, t) ((q_1 r_1 + q_{-1} r_{-1}) - (q_1 r_{-1} + q_{-1} r_1)) (p_1 + p_0 + p_{-1}).
\end{align*}
\]

From the move definitions, the sum of the probabilities of all possible three dimensional moves must be one (\(\sum_{a=1}^{1} \sum_{b=1}^{1} \sum_{c=1}^{1} p_a q_b s_c = 1\)) and from the independence of the moves in each dimension the sum in each dimension must also equal one (\(p_1 + p_0 + p_{-1} = q_1 + q_0 + q_{-1} = r_1 + r_0 + r_{-1} = 1\)). Using these identities and further constraining the moves such that there is no anisotropy along a diagonal,
\[
\begin{align*}
p_1 q_1 + p_{-1} q_{-1} &= p_1 q_{-1} + p_{-1} q_1 \\
p_1 r_1 + p_{-1} r_{-1} &= p_1 r_{-1} + p_{-1} r_1 \\
q_1 r_1 + q_{-1} r_{-1} &= q_1 r_{-1} + q_{-1} r_1,
\end{align*}
\]
we can simplify the above to,
\[
\begin{align*}
\frac{\tau}{\partial t}N(\vec{r}, t) + \frac{\tau^2}{2} \frac{\partial^2}{\partial \xi^2} N(\vec{r}, t) &= \\
(p_{-1} - p_1) \frac{\partial}{\partial x} N(\vec{r}, t) + (q_{-1} - q_1) \frac{\partial}{\partial y} N(\vec{r}, t) + (r_{-1} - r_1) \frac{\partial}{\partial z} N(\vec{r}, t) \\
+ (p_{-1} + p_1) \frac{\lambda^2}{2} \frac{\partial^2}{\partial x^2} N(\vec{r}, t) + (q_{-1} + q_1) \frac{\lambda^2}{2} \frac{\partial^2}{\partial y^2} N(\vec{r}, t) + (r_{-1} + r_1) \frac{\lambda^2}{2} \frac{\partial^2}{\partial z^2} N(\vec{r}, t).
\end{align*}
\]

If the probability of remaining at a lattice site is isotropic (\(p_0 = q_0 = r_0 = p\)) we obtain the additional identity,
\[
p_1 + p_{-1} = q_1 + q_{-1} = r_1 + r_{-1} = 1 - w,
\]
and can further simplify the lattice evolution equation to,

\[
\tau \frac{\partial}{\partial t} N(\vec{r}, t) + \frac{\tau^2}{2} \frac{\partial^2}{\partial t^2} N(\vec{r}, t) = \\
\lambda ((p_{-1} - p_1) \hat{i} + (q_{-1} - q_1) \hat{j} + (r_{-1} - r_1) \hat{k}) \cdot \nabla N(\vec{r}, t) \\
+ \frac{\lambda^2}{2} (1 - w) \nabla^2 N(\vec{r}, t).
\]

Dividing both sides by \(\tau\) and defining

\[
\vec{V} = \frac{\lambda}{\tau} ((p_{-1} - p_1) \hat{i} + (q_{-1} - q_1) \hat{j} + (r_{-1} - r_1) \hat{k}),
\]

\[
D = \frac{\lambda^2}{2\tau} (1 - w),
\]

we obtain (as \(\tau \to 0, \lambda \to 0, \) and \(\frac{\lambda^2}{\tau} \to k\)) the diffusion equation with an advection term,

\[
\frac{\partial}{\partial t} N(\vec{r}, t) = \vec{V} \cdot \nabla N(\vec{r}, t) + D \nabla^2 N(\vec{r}, t).
\]

Finally, if there is no net probability of moving in any dimension \((p_1 = p_{-1}, q_1 = q_{-1}, r_1 = r_{-1})\), then the advection term is zero and the lattice obeys the standard diffusion equation,

\[
\frac{\partial}{\partial t} N(\vec{r}, t) = D \nabla^2 N(\vec{r}, t).
\]

### 5.2.2 Multiparticle GPU implementation

**In-memory lattice representation**

The lattice is the central construct of the multiparticle model. Particles are located at uniformly spaced sites on the lattice and move from site to site according to the rules of the model. The simplest memory representation of the lattice consists of an array of memory locations, each of which stores the state of a lattice site. In the model, each lattice site’s state is defined as the number of particles of each chemical species \(\alpha\) that are located at the site. In the implementation of the model, each lattice site’s state is stored as a list containing the species of each particle at the site (an equivalent representation). Organizing the state as a single list permits more efficient storage of a sparsely populated lattice with a complex mixture of chemical species.

During a simulation, the lattice is kept in GPU global (device) memory. Lattice sites are arranged
Figure 5.2: (a) The layout of a three-dimensional lattice of size $x \times y \times z$ in memory. (b-d) The bit layout of a lattice site with a maximum of two, four, and eight particles per site, respectively. For each particle ($n_i$), the bits of the lattice site ($b_{i-j}$) that are used to store the particle’s chemical species are shown along with the bits used to store the site type.

Table 5.1: Number of particles and unique species supported by the multiparticle diffusion kernels

<table>
<thead>
<tr>
<th>Kernel</th>
<th>Number Particles</th>
<th>Bits per Particle</th>
<th>Unique Particle Types</th>
</tr>
</thead>
<tbody>
<tr>
<td>MPDiffusionModel2p</td>
<td>2</td>
<td>14</td>
<td>16383</td>
</tr>
<tr>
<td>MPDiffusionModel3p</td>
<td>3</td>
<td>9</td>
<td>511</td>
</tr>
<tr>
<td>MPDiffusionModel4p</td>
<td>4</td>
<td>7</td>
<td>127</td>
</tr>
<tr>
<td>MPDiffusionModel5p</td>
<td>5</td>
<td>5</td>
<td>31</td>
</tr>
<tr>
<td>MPDiffusionModel6p</td>
<td>6</td>
<td>4</td>
<td>15</td>
</tr>
<tr>
<td>MPDiffusionModel7p</td>
<td>7</td>
<td>4</td>
<td>15</td>
</tr>
<tr>
<td>MPDiffusionModel8p</td>
<td>8</td>
<td>3</td>
<td>7</td>
</tr>
</tbody>
</table>

In theory, two 32-bit sites could be combined to double the number of bits available for the description of particles and/or sites, but this currently has prohibitive memory and calculation ramifications. Native support...
Table 5.2: Multiparticle diffusion kernel calculation profile for a $256 \times 256 \times 256$ lattice simulation

<table>
<thead>
<tr>
<th>Calculation</th>
<th>FX5600</th>
<th>GTX280</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Time (ms)</td>
<td>%</td>
</tr>
<tr>
<td>Load lattice block</td>
<td>5.2</td>
<td>20</td>
</tr>
<tr>
<td>Random number generation‡</td>
<td>7.0</td>
<td>27</td>
</tr>
<tr>
<td>Particle movement decision</td>
<td>7.7</td>
<td>29</td>
</tr>
<tr>
<td>Particle propagation</td>
<td>3.6</td>
<td>13</td>
</tr>
<tr>
<td>Store lattice block</td>
<td>2.9</td>
<td>11</td>
</tr>
<tr>
<td>Total</td>
<td>26.4</td>
<td>100</td>
</tr>
</tbody>
</table>

†Bandwidth rates were calculated as four bytes times the number of lattice sites being transferred divided by the runtime. Operation rates were calculated as the number of logical operations per site times the number of lattice sites divided by the runtime.
‡Operation count was calculated using 64-bit operations, but current hardware implements 64-bit operations using 32-bit instructions. Performance calculated using the 32-bit instruction count (88) yields 210 GIPS and 387 GIPS, respectively.

for 64-bit values on the GPU without performance impact would enable such expanded representations.

**Processing strategy**

The time evolution of the lattice under the multiparticle diffusion model is given by the diffusion operator as shown in Equations (5.1) and (5.2). The diffusion kernel\(^1\) implements the operator by reading a lattice from global memory, calculating the position of each particle at the next time step according to the diffusion model, and then writing the new lattice back into global memory. Its general structure is as follows (see Table 5.2 for a timing profile):

1. Load a block of lattice sites from a lattice in global memory into shared memory.
2. Generate a random value for each particle’s movement.
3. Choose whether each particle should move to a neighboring site or remain in place according to the probabilities associated with the particle and site types. Store the choice for each particle in shared memory.
4. Make a list of the particles for each site that were selected to either move into the site from a neighboring site or to remain at the current site.
5. Store the list of particles at each site into a new lattice in global memory.

Since the algorithm runs in parallel on the GPU, the original lattice can not be modified until after the entire calculation has been completed. As such, the algorithm requires two separate copies of the lattice in global memory. The maximum amount of memory that can be used by a lattice is limited to one-half of the total free GPU memory. The total amount of GPU memory required for simulation of some common lattice sizes is given in Table 5.3.

\(^1\)Code destined to be executed on the GPU is organized into execution units called kernels. A kernel is compiled from C source code by the CUDA compiler into set of device specific instructions. When invoked, a kernel is downloaded to the GPU and executed in parallel using a large number of threads. Threads are organized into thread blocks, in which each thread in a block has access to a common shared memory space.
Table 5.3: GPU memory required to simulate lattices of various sizes.

<table>
<thead>
<tr>
<th>Lattice Size</th>
<th>GPU Memory Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>128x128x128</td>
<td>16 MiB</td>
</tr>
<tr>
<td>256x256x256</td>
<td>128 MiB</td>
</tr>
<tr>
<td>512x256x256</td>
<td>256 MiB</td>
</tr>
<tr>
<td>512x512x512</td>
<td>1 GiB</td>
</tr>
<tr>
<td>1024x512x512</td>
<td>2 GiB</td>
</tr>
<tr>
<td>1024x1024x1024</td>
<td>8 GiB</td>
</tr>
</tbody>
</table>

Figure 5.3: An example of the lattice sites that must be loaded into shared memory to calculate the diffusion operator on a 4x4x4 block of the lattice (from $i+j+k$ to $3\lambda i + 3\lambda j + 3\lambda k$). White sites are those in the block being processed while grey are apron sites. Symbols show (circles) the neighboring sites that must be checked for particles moving to (diamond) a site being processed.

**Shared memory constraints**

The first step of the diffusion algorithm loads a block of the lattice into shared (on-chip) memory. From shared memory, the lattice sites can be processed by a thread block (one thread per lattice site) without the latency and bandwidth limitations associated with global memory. Significantly, not only are the lattice sites being processed loaded into shared memory, but also an apron of sites surrounding them. These additional sites are required because particles may enter a lattice site from any of its nearest neighbors. Even though an apron site will not be stored into the final lattice by the thread block, the movement of its particles must still be calculated so that it can be determined which of the particles (if any) move into a lattice site that is being processed by the thread block.

As can be seen from Figure 5.3, when a small three-dimensional block of lattice sites is being processed the apron sites account for a large fraction of the total sites. The amount of shared memory required to process a cubic block of lattice sites of dimensions $B \times B \times B$ is $4 \cdot (B + 2)^3$ bytes with the memory used for the apron sites being $4 \cdot ((B + 2)^3 - B^3) = O(B^3)$ bytes (Figure 5.4 (a)). The total available shared memory on current generation GPUs is 16 KB per multiprocessor (minus a small amount of overhead). The largest block that can be loaded has thirteen lattice sites per side ($4 \cdot (13 + 2)^3 = 13500$). In order for the GPU to efficiently process thread blocks, though, the actual amount of shared memory dedicated to a single thread block must only be a fraction of the total. For a block size of eight sites, the shared memory required is 3.9 KB (still a somewhat high allocation), of which roughly half is needed for apron sites.

However, shared memory usage is not the only overhead associated with the apron sites. Since apron
sites are only processed by a thread block to find the particles moving from them, the sites must also be processed again to determine the particles moving to them (when the site is processed as part of another block). The diffusion operator must be calculated twice for these sites. Figure 5.4 (b) shows the increase in the number of diffusion calculations required as a result of the apron calculations. The number of calculations that must be performed for a block size of eight sites per side is twice the total number of lattice sites. Half of all the diffusion calculations would be redundant calculations required for the apron sites.

The combined effect of the shared memory and calculation efficiency requirements, makes calculating the diffusion operator computationally expensive for a three-dimensional block of lattice sites. Smaller blocks are less efficient to calculate (large surface area to volume ratio), but larger blocks cannot be processed because of shared memory limitations. One technique to alleviate this contradictory condition is to remove the square dependence of the number of apron sites on the block size. In Section 5.2.1, it was shown that the three-dimensional diffusion model can be equivalently expressed as a decomposition into each dimension independently. Implementing the diffusion operator as three successive diffusion calculations, one in each dimension, dramatically decreases the number of apron sites that must be loaded and redundantly calculated. Lattice sites are processed in a one-dimensional block of length $B$. The shared memory necessary to process a block is $4 \cdot (B + 2)$ and the memory used by the apron sites is $4 \cdot ((B + 2) - B) = 8$ bytes (constant regardless of the block size). For a one-dimensional block size of 32 lattice sites and larger there is only minimal overhead associated with the apron sites (a 6% increase in shared memory usage and required calculations).

As can be seen from Table 5.2, the transfer rate when loading and storing lattice sites to and from shared memory, is short of the theoretical maximum value (80 GB/s for FX5600 and 140 GB/s for GTX280). When loading the lattice, the apron sites require that some threads perform multiple loads (during which the other threads are idle), accounting for a portion of the shortfall. However, there still appears to be additional overhead, possibly related to GPU occupancy being sub-maximal due to the number of registers required by the monolithic diffusion kernel. Additional analysis may lead to further improvement in the rates.

![Graphs](image)

Figure 5.4: (a) Shared memory usage for the diffusion operator as a function of the size of the block being processed. The memory used for (blue) all sites and (red) apron sites are shown for comparison. (b) The calculation overhead due to the apron sites.
Figure 5.5: Pattern of mapping thread blocks to lattice sites for x, y, and z diffusion kernels. White sites are those being processed by the thread block and grey sites are the apron sites required for the calculation. In the y and z diffusion kernels, sixteen one-dimensional blocks of sites are processed by each thread block such that all threads of a half-warp read sequential 32-bit words from global memory.

Thread block definition

Calculating diffusion in each dimension independently requires three separate kernel invocations, one for each dimension. During an invocation, each thread block processes a block of the lattice according to the thread block patterns shown in Figure 5.5. In the x dimension, a thread block maps to a one dimensional block of lattice sites. In the y, and z dimensions, however, the thread blocks map to sixteen such blocks corresponding to slices of the lattice in the x-y and x-z planes. These thread block definitions ensure that accesses to global memory are always performed in 64-byte contiguous reads, which can be coalesced by the GPU for maximum memory throughput. Each lattice site is processed by an individual thread in the thread block and the 32-bit size of each site ensures that there are no conflicts when accessing shared memory.

In the x dimension, a block size of 128 sites gives the highest performance. In the y and z dimensions, the optimal size is 16x8. Interestingly the loading of lattice sites from global memory performs differently in the y and z dimensions. Loading from the x-y plane performs better than loading from the x-z plane. The primary difference between these two methods is the locality of the memory addresses and we attribute the performance difference to misses or conflicts in the GPU global memory caches or translation lookaside buffer (TLB). The properties of the GPU memory system, including the cache and TLB sizes and latencies, have recently been thoroughly investigated in [247].

Random number generation

A large fraction (∼25%) of the calculation time required for the multiparticle diffusion model is spent generating random numbers. For a time step, each particle requires three random numbers (one for each dimension) to realize its movement according to the site transition probabilities. To generate the random values, we use a combination of 64-bit random generators, as described in Press et al. [248]. Specifically, we use a linear congruential generator, followed by a 64-bit xorshift, and finally a pass through a multiplicative linear congruential generator.

Random number generation is constrained by the requirement that particles that fall into apron sites must
have their diffusive motion calculated multiple times, each time returning the same result. To enforce the constraint the random value generated for each particle is a random hash based on a 128-bit value containing the site index, particle index, and time step. The combination guarantees a unique but reproducible random value for each particle at each site for each timestep. Additionally, a seed value must be specified to make each simulation a unique realization of the model, which is also incorporated into the calculation.

Any correlations produced by the random number generator would cause the simulation results to deviate from the true distribution of the underlying model. We have checked the generator for such using the “BigCrush” test suite from the TestU01 random number testing library [249]. The method passed all tests.

**Constant memory transition probabilities**

In order to determine a particle’s motion at a time step, the diffusion kernel makes a random choice based on the transition probabilities for moving to the site in the minus direction, staying at the current site, or moving to the site in the plus direction. In general, the transition probability for a particle moving from one site to another depends on the particle type, the source site type, and the destination site type. This granularity of transition probabilities permits very flexible spatial geometries to be simulated, including regions of the lattice where different diffusion conditions apply on a per-particle basis. The memory required to store the transition probabilities, though, can become significant when multiple site and particle type transitions are defined. Since these probability tables remain constant throughout a kernel invocation and are identical for each thread block, they can be stored in the GPU’s constant memory space instead of occupying additional space in shared memory.

Constant memory is read-only memory that, when cached, performs at register speed. Each multiprocessor on the GPU has 8 KB of constant memory cache, so if the probability table is less than that size it can fit entirely into the cache. The probability tables for a typical system with four unique site types and ten different particle types would occupy 0.7 KB. Constant memory performs optimally when each thread in a warp accesses the same location, otherwise it scales linearly [244]. In general, each particle and lattice site being processed in a warp will not be the same, so the theoretical maximum constant memory bandwidth will not be achieved. However, in a typical simulation, most nearby lattice sites will be of the same type and very few particle types will be present in any one region of the lattice. Only around 0.06% of the constant memory reads are expected to diverge within a thread warp.

**Boundary condition processing**

Since the multiparticle model is lattice-based, it must deal with boundary conditions when a site is on the lattice edge. Three boundary conditions are commonly used in lattice models: reflecting, absorbing, and periodic. The implementation of the diffusion kernel makes processing reflective or absorptive conditions straightforward. If a lattice block being processed is on the lattice edge, its off-lattice apron sites are not loaded from global memory but instead are set to a specific “boundary” site type. The diffusion transition probabilities for a particle to move from any site type to the boundary site type are then set appropriately. For reflective boundary conditions, the transition probability is zero from any site to a boundary site, equivalent to a particle bouncing off the boundary and returning to the original site. In the case of absorptive boundary
conditions, the boundary transition probability is equivalent to the transition rate for the particle to a site of the same type it currently resides in. If a particle transitions into a boundary site it becomes off-lattice and is destroyed (not placed into the new lattice) by the normal diffusion calculation, accomplishing the absorption. As a side effect of defining a unique boundary site type, the effective number of sites types that can be used in a simulation is reduced from eight to seven, but the advantage is practically no performance penalty for boundary condition processing.

Periodic boundary conditions are implemented by loading off-lattice apron sites from global memory corresponding to the opposite side of the lattice. The extra global memory reads compared to reflective or absorptive conditions means that periodic boundary conditions suffer a performance penalty, which in practice is around 10%. Diffusion processing after the apron load follows the normal calculation path, no other special processing is required for periodic boundaries.

Site overflow

In the multiparticle model an unlimited number of particles can theoretically be located at a lattice site. In the implementation, though, there are a limited number of bits available for storing particles at each lattice site. Three separate kernels are implemented allowing a maximum of 2, 4, or 8 particles at each site. If, during an calculation, more than this number of particles are moved to a site the kernel must gracefully handle the overflow; particles cannot be lost. To avoid losing particles we use an overflow list, a list (stored in global memory) of all the sites that overflowed during a diffusion calculation. If the kernel detects that a lattice site has more particles than the maximum allowable, it stores the index of the lattice site in the overflow list along with the chemical identities of the extra particles. After each diffusion kernel execution, the extra particles from every site in the overflow list are randomly placed back into the lattice at a nearby site of the same type (done in CPU code). The overflow list prevents particle loss, but using it incurs a computational cost. To achieve optimal performance, simulation parameters should be chosen such that sites rarely overflow. The overflow list then becomes an exception mechanism to handle low frequency events.

In order to choose the appropriate parameters to avoid excessive sites overflows during a simulation, one must first know the chance of a site overflow occurring for the lattice configuration. As an estimation of the probability during a simulation, consider the process of adding \(N\) particles to an empty lattice \(L\) with \(L_s\) total sites. Assuming that all sites are equally probable, the probability of placing a particle at any given site is \(\frac{1}{L_s}\). The probability of a site containing \(n\) particles after all \(N\) have been added \((p(n))\) is therefore the probability of placing \(n\) particles into the site during \(N\) independent choices. This probability is given by the binomial distribution,

\[
p(n) = \frac{N!}{n!(N-n)!} \left(\frac{1}{L_s}\right)^n \left(1 - \frac{1}{L_s}\right)^{N-n},
\]

\[
= \binom{N}{n} \left(\frac{1}{L_s}\right)^n \left(1 - \frac{1}{L_s}\right)^{N-n}.
\]  

(5.5)

\(2\)Kernels running on GPU hardware lacking atomic functions (pre 1.1 compute capability) may still lose particles if more than one thread simultaneously writes to the overflow list.
Multiplying the probability for a single site by the total number of sites we get the expected number of lattice sites having exactly \( n \) particles,

\[
E(n) = L_s p(n).
\]

Finally, the expected number of lattice sites that will overflow, i.e., exceed the maximum number of particles that can be stored in a site \( (n_{\text{max}}) \), is

\[
E(n > n_{\text{max}}) = L_s \left( 1 - \sum_{i=0}^{n_{\text{max}}} p(i) \right). \tag{5.6}
\]

From Equations (5.5) and (5.6) it can be seen that the expected number of sites that will overflow depends on both the number of particles on the lattice and the lattice size. Although, for two lattices of different sizes the probability of a site overflowing is approximately equal if their occupancy (mean number of particles per site) is the same, there are more sites in a larger lattice that can overflow so the expected number of overflows is larger.

For a simulation to run as efficiently as possible, the number of particles on the lattice must be such that
site overflows happens infrequently, perhaps once in every one hundred time steps. Approximating each
timestep as an independent lattice configuration, the maximum number of particle per site and the lattice
occupancy should be chosen such that the expected number of site overflows is \( \leq 0.01 \). Figure 5.6 (a)
shows the expected number of site overflows as a function of the lattice occupancy. For a 256\( \times \)256\( \times \)256
lattice, the maximum allowable occupancy is 0.0015, 0.038, and 0.41 particles per site for \( n_{\text{max}} = 2, 4, \) and 8, respectively. In general, it is also useful to interpret the lattice occupancy as a concentration. The
concentrations corresponding to the maximum occupancy (such that \( E(n > n_{\text{max}}) \leq 0.01 \)) are shown in
Figure 5.6 (b) as a function of lattice spacing. For a 256\( \times \)256\( \times \)256 lattice with 1 nm spacing, the maximum
particle concentration is around 2 mM for \( n_{\text{max}} = 2 \), 60 mM for \( n_{\text{max}} = 4 \), and 650 mM for \( n_{\text{max}} = 8 \). At
smaller lattice spacing, care must also be taken so that the maximum concentration does not exceed what is
physically realistic.

While the techniques described above minimize the overhead of overflow handling, they do not eliminate
it. Control must still be returned to the calling program on the CPU after each kernel execution to check
for overflow exceptions, incurring a \( \sim 1 \) ms overhead for each kernel invocation. It may be possible to
improve performance by implementing an entirely GPU based exception mechanism following the global
GPU barrier technique introduced in [247]. In general, exception handling techniques are still an under-
developed area of GPU programming.

**Conditional compilation**

A final GPU programming strategy that provided significant performance improvements was to factor out a
conditional check from the diffusion kernel when it remained constant during the entire invocation. While
seemingly trivial, simple “if” statements in a kernel can have a significant impact on overall execution time,
even if they evaluate identically during an invocation. By refactoring a single combined kernel into two
kernels, one in which the condition is assumed true and one in which it is assumed false, and then invoking
the appropriate kernel at runtime, one can move the comparison logic off on the GPU, where it must be
evaluated by each thread, and on to the central processing unit (CPU), where it must be performed only
once.

In our multiparticle GPU implementation, we factored out the number-of-particles-per-site conditional
check and the boundary-condition check, creating separate kernels for each. These improvements provided
around a 5% performance increase compared to the combined kernel. We also created a separate kernel
that is called when only a single diffusion coefficient has been specified. This simplification eliminates the
constant memory lookup for simulations in which all particles diffuse at the same rate and provides a large
performance enhancement (10% speedup) for that common use case.

**5.2.3 Analysis of diffusion simulations**

**Free diffusion**

To validate the implementation of the multiparticle model, we first assessed its characteristics when simu-
lating freely diffusing particles on a periodic lattice. Under such conditions, the particle distributions should
Figure 5.7: Results from a simulation of 16,380 particles undergoing free diffusion with a diffusion coefficient (D) of 10 nm²/µs on a periodic lattice with spacing 2 nm and a time step of 10 ns. (left) Agreement of RMSD, MSD, and D between (blue solid) the simulation data and (red dotted) the expected values calculated using the multiparticle diffusion model. (right) Displacement probability in the three principle dimensions at four time points in the simulation. Shown are (circles) the probabilities as calculated from the simulation, (colored solid) the non-linear least squares fit of the simulation probabilities to a normal distribution with the indicated mean and standard deviation, and (black dotted) the expected normal distribution according to the multiparticle diffusion model.

agree with the continuum model presented earlier. For a particle undergoing Brownian diffusion, the relationship between its MSD and the amount of time it has been freely diffusing is given by the well-known relation, in three dimensions, \(< r^2 > = 6Dt \) [250]. Additionally, the complete probability distribution in each dimension is given by \( \frac{1}{2\sqrt{\pi}Dt} e^{-x^2/4Dt} \), which is a Gaussian distribution with variance \( \sigma = \sqrt{2Dt} \).

We performed 10 ms simulations of 16,380 particles with nine different diffusion coefficients (200, 100, 50, 25, 10, 5, 1, 0.1, and 0.01 nm²/µs) freely diffusing on a periodic 128×128×128 lattice. The lattice’s natural diffusion coefficient was the same in each simulation (200 nm²/µs; 2 nm spacing, 10 ns time step) and only the transition probabilities varied (\( p_{-1} = p_1 = 0.5, 0.25, 0.125, 0.0625, 0.025, 0.0125, 2.5\times10^{-3}, 2.5\times10^{-4}, 2.5\times10^{-5} \)). The calculated and expected values for MSD and D agree over the entire course of each simulation (see Figures 5.7 and 5.8). For simulations with large diffusion coefficients...
(relative to the lattice’s natural D), the agreement of the particle distribution with a Gaussian, as judged by the Jarque-Bera test value [251], is also within the expected range. The simulations with the two smallest diffusion coefficients (0.1 and 0.01 nm²/µs), however, show initial deviation from a normal distribution. This situation is not unexpected as for these two simulations the number of particles moving to a new lattice site at each time step are approximately ten and one, respectively. Since lattice jumps are of a discrete size, the particle distributions in these simulations only approximate a Gaussian after a significant number of time steps. After 2 ms of simulation, even diffusion coefficients four orders of magnitude lower than the lattice’s natural value are reasonable approximations to the continuum model.

**Constant concentration boundary conditions**

Constant concentration boundary conditions connect a simulated volume with an infinite reservoir of particles at a given concentration. Since the constant concentration begins at the edge of the simulation space and particles diffuse at a finite speed, a time dependent concentration gradient will exist within the volume when the internal and external concentrations are not equal. The time behavior of this concentration gradient is a unique feature of spatial simulations that cannot be approximated in well-stirred models. Therefore, it is important to ensure that the correct time evolution is produced by constant concentration boundaries in the multiparticle model.

The time and space evolution of a concentration of particles in one dimension, \( c(x, t) \), is given by the
Figure 5.9: Time dependent concentration gradient along the x dimension during a simulation of an initially empty lattice connected to a 1 µM constant concentration reservoir. To model one-dimensional diffusion, only the x dimension had constant concentration boundaries, the y and z dimensions were periodic. (colored solid) The average over ten lattice simulations and (black dotted) the solution to the one-dimensional diffusion equation are shown for four values of the diffusion coefficient (D).

A one-dimensional version of the diffusion equation,

\[
\frac{\partial}{\partial t} c(x, t) = D \frac{\partial^2}{\partial x^2} c(x, t),
\]

where D is the diffusion coefficient. To solve this equation for a given domain, \(0 \leq x \leq L\), the boundary conditions at \(x = 0\) and \(x = L\) must be specified. Setting \(c(0, t) = c(L, t) = C_0\), where \(C_0\) is some constant concentration, and \(c(x, 0) = 0\) everywhere except at \(x = 0\) and \(x = L\), one can evaluate how the particle flux through the boundaries changes the concentration of an initially empty system. The one-dimensional diffusion equation can be quickly solved numerically using an explicit finite-difference method. Although simple, this method is known to have convergence for the linear diffusion equation as long as the time (\(\delta t\)) and length (\(\delta x\)) steps obey the relation \(\frac{\delta t}{(\delta x)^2} \leq \frac{1}{2}\).

An identical system can be constructed using the multiparticle model for comparison by starting with an empty lattice and using constant concentration boundaries in the x dimension and periodic boundaries in the y and z dimensions. Figure 5.9 shows the mean concentration profile along the x dimension of such a lattice at various times. The lattice size was 256 nm x 256 nm x 256 nm with 2 nm spacing and the simulation was performed with 10 ns time steps. Ten identical replicates were performed for each of four values of D. The concentration at each x position was compared against the expected concentration from the solution of the one-dimensional diffusion equation and was found to have an average relative error of <5%. Most of the deviation is due fluctuations resulting from the discrete, stochastic nature of the multiparticle model and the
values would further converge with additional replicates. Overall, there is excellent agreement between the two models.

**Obstructed diffusion**

To test our model’s ability to reproduce anomalous subdiffusion in crowded environments, we constructed a periodic lattice with obstructions modeled as clusters of reflective lattice sites. Clusters were determined by mapping a sphere with the diameter of the obstruction onto the lattice and setting each site located within the sphere as reflective. Particles diffusing on the lattice have zero probability to transition to a reflective lattice site, and must diffuse around the obstacles. In this approximation lattice obstructions are stationary, a reasonable assumption for larger obstacles but which is less realistic as the diameter of an obstacle approaches the lattice spacing.

Simulations with obstacle sizes ranging in radius from 100 nm to 1 nm were simulated at three different obstructed volume fractions (10%, 20%, and 30% by volume). Analysis of the simulations shows that the multiparticle lattice model does exhibit anomalous subdiffusion in crowded systems (Figure 5.10). Like other computational models of crowded diffusion, it shows normal Brownian diffusion at short times, a cross-over period during which diffusion is anomalously subdiffusive, and finally a return to normal diffusion at long times. These phases can be respectively understood as the time during which a particle is able to freely diffuse without encountering an obstacle, the time period in which a particle’s motion is affected by a single obstruction, and the time in which all particles have encountered many obstructions. As the size of the obstruction increases, the crossover occurs at large time windows; larger obstacles will have greater mean spacing at a given volume fraction than smaller obstacles. For an obstructed volume of 30%, the crossover time with obstacles 20 nm in diameter is in the microsecond range. This result is particularly relevant as 20 nm is approximately the diameter of a ribosome, one of the most abundant large (compared to a protein) particles in the cell, occupying 8-10% of the volume.
In vivo modeling

The in vivo cytoplasm is more complex than can be modeled by obstacles of a single size. Ridgway et al. [225] used the available proteomics data to describe the cytoplasmic environment of an E. coli cell in terms of the populations of different size classes of particles. We used the same particle classes and populations to construct a lattice model of a stationary in vivo environment. The technique described above was used to map particles to lattice obstructions. Figure 5.11 shows the lattice representations of the particle classes along with illustrations of a periodic volume used for analyzing diffusion under the model and a full-size E. coli cell. We simulated diffusion of particles with various diffusion coefficients in the in vivo environment to test the effects of the stationary obstruction and lattice approximations (see Figure 5.12). The in vivo diffusion of proteins is reduced by approximately 20% in the simulations (smaller decreases were observed for particles with lower diffusion coefficients). This is somewhat less than the 30% reduction seen in the

Figure 5.11: (a) Lattice representations of in vivo obstacles with the indicated diameter and their abundances in the in vivo environment. (b) 256 nm × 256 nm × 256 nm lattice modeling an in vivo environment (30% obstructed volume). Black points are tracer proteins (1 nm) in diameter diffusing in the free volume. Offset are 1 nm and 5 nm thick slices through the lattice. (c) An in vivo packed representation of an E. coli bacterium, 2 μm in length and 0.8 μm in diameter.

Figure 5.12: The effect of particles diffusing in a periodic volume containing an approximated in vivo environment occupying 30% of the volume. Shown is the apparent diffusion coefficient in an in vivo environment of a particle having the indicated in vitro diffusion coefficient.
Table 5.4: Whole cell GPU performance

<table>
<thead>
<tr>
<th>Lattice Size</th>
<th>Spacing (nm)</th>
<th>Time Step (µs)</th>
<th>Calculation Perf. (10⁶ sites/s)</th>
<th>Simulation Perf. (s/GPU·day)</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>FX5600/GTX280</td>
<td>FX5600/GTX280</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>64²×128</td>
<td>20</td>
<td>8.00</td>
<td>219/533</td>
<td>290/700</td>
<td>2.4X</td>
</tr>
<tr>
<td>64²×128</td>
<td>16</td>
<td>5.12</td>
<td>212/522</td>
<td>180/440</td>
<td>2.4X</td>
</tr>
<tr>
<td>128²×256</td>
<td>10</td>
<td>2.00</td>
<td>310/781</td>
<td>13/32</td>
<td>2.5X</td>
</tr>
<tr>
<td>128²×256</td>
<td>9</td>
<td>1.62</td>
<td>307/747</td>
<td>10/25</td>
<td>2.5X</td>
</tr>
<tr>
<td>128²×256</td>
<td>8</td>
<td>1.28</td>
<td>302/776</td>
<td>8.0/20</td>
<td>2.5X</td>
</tr>
<tr>
<td>256²×512</td>
<td>7</td>
<td>0.94</td>
<td>349/648</td>
<td>0.85/1.6</td>
<td>1.8X</td>
</tr>
<tr>
<td>256²×512</td>
<td>6</td>
<td>0.72</td>
<td>348/647</td>
<td>0.65/1.2</td>
<td>1.8X</td>
</tr>
<tr>
<td>256²×512</td>
<td>5</td>
<td>0.50</td>
<td>347/645</td>
<td>0.45/0.83</td>
<td>1.8X</td>
</tr>
<tr>
<td>256²×512</td>
<td>4</td>
<td>0.32</td>
<td>346/642</td>
<td>0.29/0.52</td>
<td>1.8X</td>
</tr>
</tbody>
</table>

Brownian dynamics models, where obstructions are mobile. However, we see the same crossover time scale, during which diffusion is anomalous (10⁻⁶ s).

Despite the approximations involved, the lattice model appears to capture the intrinsic nature of the effect of in vivo crowding on diffusion. Unlike Brownian dynamics models, where performance scales with the number of particles, the multiparticle diffusion model described here is invariant toward the number of obstacles. Its performance depends only on the total number of lattice sites. Using a single GPU, in vivo simulations can extends well into the seconds time range (see Table 5.4). The effects of the lattice discretization and non-mobile obstacles, while significant, can often be justified by the large increase in simulation time compared to other simulation methods. The extent of the lattice discretization effects can be controlled by adjusting the lattice spacing and time step for the specific problems being addressed.

5.3 Reaction Kinetics

5.3.1 Reaction model

In addition to diffusively moving on the lattice, particles are able to react with each other according to defined stoichiometry and kinetic rates. We assume a well-stirred environment in each lattice site during a timestep and calculate the probability of a reaction occurring in a Gillespie-like manner [252]. The reaction operator (R) is applied following the diffusion operator and updates a lattice site to account for any reactions during a timestep,

\[ N_\alpha(\vec{r}, t + \tau) = \mathcal{R} \cdot \mathcal{D} N_\alpha(\vec{r}, t). \]

It calculates a random realization of all the possible reactions given the current state of a lattice site and the stoichiometry matrix (S), which contains the changes in the counts of the chemical species for each of the M reactions:

\[ \mathcal{R} N_\alpha(\vec{r}, t) = N_\alpha(\vec{r}, t) + \sum_{m=1}^{M} S_{m,\alpha} \theta(m, N(\vec{r}, t), t), \quad (5.7) \]
where the function $\theta(m, \vec{N}, t)$ is a stochastic function returning 1 if reaction $m$ occurs in lattice site $N$ at time $t$, otherwise 0. It is a function such that the probability of a reaction occurring is consistent with

$$P(\theta(m, \vec{N}, t) = 1) = \int_0^T a_m(\vec{N}) e^{-\sum_m a_m(\vec{N}) t'} dt'.$$

$a_m(\vec{N})$ is the Gillespie propensity (probability per unit time) of reaction $m$ occurring given a quantity of reactants.

5.3.2 Reaction GPU implementation

Processing strategy

Reaction processing as shown in Equation 5.7 is performed by the reaction kernel. The kernel reads a lattice site from global memory, updates the site according to the reaction operator, and then writes the updated site back to global memory. The overall algorithm is as follows:

1. Load a lattice site from global memory.
2. Unpack the particles in the site into individual variables.
3. Sum the reaction propensity for each reaction, using the particles at the site, to calculate the total reaction propensity. Reaction parameters are stored in constant memory.
4. Calculate the probability of a reaction occurring in the site.
5. Randomly decide if a reaction occurred using the reaction probability.
6. If a reaction occurred, randomly choose which one according to the individual reaction propensities.
7. Update the particle counts according to the reaction.
8. Write the particle counts back to global memory.

As opposed to the diffusion kernel, the reaction kernel site depends only on the particles in that site and affects only the same site. Therefore, shared memory is not used. A lattice site is loaded directly into the registers of the thread that is processing the site.

Single precision effects on probability calculation

During a timestep the probability of a reaction occurring is calculated independently for each lattice site based on the number and species of the particles located at the site. The probability is calculated according to $p = 1 - e^{-\lambda}$, where $\lambda$ is the total reaction propensity for the site. Calculating an accurate probability is critical to correctly sampling the dynamics of the system. Unfortunately, calculating $p$ using single precision arithmetic, which GPUs are optimized to calculate, yields an increasingly inaccurate value for $\lambda \leq 1 \cdot 10^{-4}$.
Figure 5.13: Relative error during the calculation of a reaction probability using the GPU as a function of the rate constant \( \lambda \). The probability is calculated as \( p = 1 - e^{-\lambda} \). Three methods of calculation are compared: (blue) double precision, (red) single precision, and (green) a switching function that uses the approximation \( 1 - e^{-\lambda} \approx \lambda \) for \( \lambda \leq 2 \cdot 10^{-4} \).

That becomes truncated to zero around \( 1 \cdot 10^{-7} \) (see Figure 5.13). Many rates in systems of biological interest fall within this range necessitating an alternative calculation method.

To work around the GPU single precision limitation, we use one of two techniques. For GPUs supporting CUDA compute capability 1.3 or higher, we use double precision arithmetic, which is supported in hardware on these GPUs, albeit at a significantly decreased speed. To reduce the performance impact we only perform the probability calculation in double precision:

```c
__device__ float calculateReactionProbability(const float rate)
{
    return (float)(1.0-exp(-(double)rate));
}
```

all other calculations in the reaction kernel are performed using single precision arithmetic on CUDA 1.3 devices.

For GPUs lacking double precision hardware, we instead implement a switching function that uses the approximation \( 1 - e^{-\lambda} \approx \lambda \) for \( \lambda \leq 2 \cdot 10^{-4} \). This is simply the first two terms of a Taylor expansion of \( e^x \) around 0 (\( e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!} \)),

\[
1 - e^{-\lambda} = 1 - \sum_{n=0}^{\infty} \frac{(-\lambda)^n}{n!},
\]

\[
\approx 1 - (1 + (-\lambda)),
\]

\[
\approx \lambda.
\]

As can be seen from Figure 5.13, the relative error of the single precision switched form of the probability...
Table 5.5: Relative performance of probability calculation methods on the GPU

<table>
<thead>
<tr>
<th>Method</th>
<th>GeForce 8800 GT</th>
<th>Tesla C1060</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single Precision</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>Switching Function</td>
<td>1.25</td>
<td>1.4</td>
</tr>
<tr>
<td>Double Precision</td>
<td>n/a</td>
<td>27.9</td>
</tr>
</tbody>
</table>

calculation has decreased accuracy compared with the double precision calculation, but still never exceeds 0.0001. There is, however, also a performance impact associated with the switched form of the function, as the rate must be compared to the cutoff rate and different threads in the same warp may diverge if their rate arguments are on opposite sides of the cutoff.

To measure the performance impact of using either double precision for key calculations or a single precision switching function, we implemented timing kernels for both cases along with a baseline kernel performing solely the single precision calculation. These timing kernels perform many simultaneous replicates of the calculation being investigated and represent the maximum theoretical performance difference between the calculation methods. As can be seen from Table 5.5, the switching function incurs a calculation overhead of 25–40% and the overhead for the double precision calculation on supported hardware is nearly 2800%. In practice, much of the performance impact is hidden by memory latency and using double precision calculations appears to produce only a limited slowdown under realistic simulation conditions. Even so, the theoretical results provide useful guidance for evaluating the different calculation methods, especially for the choice to limit double precision calculations to the minimum set necessary to achieve the desired accuracy.

Finite precision effects on random number distribution

Calculation of random numbers results in a 32-bit value, which is then mapped to [0.0 1.0) for comparison against the total reaction probability. This mapping yield a discrete (non-continuous) series of probabilities separated by $1/2^{32}$. Number between these steps can never be generated, so comparison to a floating point probability yields an error on the order of the step size. When comparing to very small probabilities the relative error can be significant, particularly once the probability is near the step size. In particular, since random values below $1/2^{32}$ never occur, probabilities below this value occur never occur. Figure 5.14 shows the effect of this finite precision on random number generation.

Because of the increasing error below a certain probability, we set a lower limit for valid probabilities such that the numerical error is always ≤ 1%, which corresponds to a probability of $1.141 \times 10^{-8}$. Using this minimum probability, one can determine the minimum kinetic rate that can be simulated for a given lattice. The parameter tables given in Appendix B show the lower limits for first and second order reactions for a variety of common lattice configurations.
Figure 5.14: (top) Probability of obtaining a given value when performing a uniform random sampling of the range [0.0 1.0) using 32-bit precision. (middle) Cumulative probability of obtaining a value less than or equal to a given value. (bottom) Relative error in the cumulative probability. Vertical line marks the probability above which the relative error is always $\leq 1\%$.

**Single reaction per site limitation**

In the reaction model presented above, only a single reaction can occur per lattice site per time step. This processing method avoids the expense of a looping construct that would select all possible reactions that would occur during the time step, essentially implementing the Gillespie direct method in each lattice site. However, if it is likely that multiple reactions would actually have occurred, the simplification would introduce significant errors in the kinetics. To avoid this, we ensure that the probability of any one reaction occurring in a given lattice site during a time step is always $\leq 1\%$. The parameter tables given in Appendix B show the maximum kinetic rates that correspond to this limit for a variety of lattice configurations.
Chapter 6

Spatial Noise in the Lactose Utilization Genetic Circuit

6.1 Background

6.1.1 Stochasticity in gene regulation

Transcription networks control the phenotype of modern cells, regulating the expression of proteins according to a genetic program. These networks are responsible for such varied processes as the regulation of gene expression in response to changing environmental conditions and/or external stimuli and the differentiation of cells along developmental pathways. It has been well established that a limit on the accuracy of gene regulation exists due to intrinsic noise caused by the discrete biochemical nature of the process [253]. On top of this, there is also an extrinsic component to the total noise arising from random cell-to-cell variability in the number of copies of the transcription and translation machinery (transcription factors, RNA polymerases, ribosomes, etc) [254–256]. In fluctuating environments, this stochastic noise can lead to different phenotypic outcomes for initially identical cells [257–259] and, for certain systems, the resulting heterogeneous population can be more optimal for growth than would be a population containing a single phenotype [260].

An additional influence on stochastic noise in biochemical systems that has recently come under scrutiny is due to spatial heterogeneity within a cell and molecular crowding in the in vivo environment; it is becoming increasingly apparent that the cell is not a well-stirred system [261–263]. Studies using cryo-electron tomography techniques [264–266] have revealed that individual macromolecules are not necessarily uniformly distributed inside the cell, but may be clustered in a spatially dependent manner. Spatial organization can affect reaction kinetics by increasing local concentrations of reactants and enzymes. Additionally, crowding and non-specific molecular interactions in the in vivo environment can lead to anomalous (sub) diffusive behavior for macromolecules.

Theoretical modeling of stochasticity in gene expression has a long history, starting with early studies of the phage lambda decision circuit [267; 268], and has greatly increased our understanding of the effect of noise on gene expression (for a review see [269]). Computational modeling may be the only tractable way to evaluate very complex genetic systems; in some cases these systems must be evaluated sequentially in time.

The contents of this chapter are based in part on a manuscript in preparation by Elijah Roberts, Andrew Magis, and Zaida Luthey-Schulten.
to discover the emergent behavior they produce. If models are unable to account for spatial heterogeneity, systems where spatial effects are important (*e.g.*, developmental pathways of multicellular organisms) will be inaccessible to computational biology. Recently, several such modeling efforts have been undertaken [233; 270–274].

Stochastic modeling, however, is not a panacea for studying gene expression networks. Painstaking effort must be undertaken to obtain parameters for the models. Parameters, which are often obtained under *in vitro* conditions, must be validated by comparing modeling results to published experiments. Recently, time-lapse fluorescence microscopy has been used to reveal dynamic behavior for individual macromolecules *in vivo* [275; 276]. Parameters obtained from single molecule experiments are uniquely suited for stochastic modeling, as they provide true distributions not simply mean values from ensemble measurements. Equally importantly they can be measured under *in vivo* conditions and represent the true cellular environment [222]. Also, high resolution imaging studies can provide spatial information [277; 278].

Theoretical studies of the stochastic behavior of genetic circuits have shown that populations of cells can be quite heterogeneous, even starting from an initially identical state [279–281]. The large variance in population distribution is predicted to come about do to bursting in the process of gene transcription. When bursts are infrequent relative to the lifetime of the protein and the bursts are independently distributed, the population will migrate toward a negative binomial distribution (a gamma distribution in the continuous approximation) in the steady state. Recent *in vivo* single-molecule fluorescence studies have observed this distribution in LacY expression of living cell populations, and furthermore, have shown a linear relationship between the inferred burst size and the inducer concentration [258]. Since previous theoretical studies have only considered transcriptional bursting of static genetic models, we wanted to investigate the stochastic noise effects of an inducible genetic system.

### 6.1.2 Lactose uptake in *Escherichia coli*

The lactose utilization system in *E. coli* is a model system for studying inducible genetic circuits [282–288]. The overall genetic circuit is illustrated in Figure 6.1. Briefly, *lac* repressor (LacI) binds to the operator region upstream of the DNA encoding for the genes responsible for lactose uptake and metabolism, repressing their expression in the absence of lactose. In the presence of lactose (or other inducer), LacI binds the inducer preferentially and is prevented from binding to the operator region allowing expression of the proteins in the *lac* operon. One protein in the operon, lactose permease (LacY), establishes positive feedback in the circuit by inserting into the membrane and transporting lactose into the cell, ensuring that LacI remains sequestered.

Many theoretical and experimental studies have investigated the behavior of *lac* system and shown it to be stochastic, depending on random fluctuations to switch between the off and on states. Models of stochastic chemical systems are usually expressed in terms of the chemical master equation (CME), which describes the time evolution of the probability of a system to be in a given state [289], and often computationally analyzed used a variant of the Gillespie’s stochastic simulation algorithm [252; 290]. The CME, however, assumes that the system is well-stirred. To investigate possible spatial effects on the *lac* system, we modeled it using our lattice microbe method. We took parameters preferentially from (or fit
Figure 6.1: Overview of the *lac* genetic circuit in *E. coli*. (a) In the absence of inducer, the lac repressor (LacI) binds to (red) the *lac* operator preventing transcription of genes in (yellow) the *lac* operon. (b) Following an increase in the extracellular inducer concentration, inducer enters the cell via both diffusion across the membrane and active transport by lactose permease (LacY). Once inside, inducer binds free LacI molecules preventing them from binding to the operator. (c) Once the intracellular inducer concentration reaches a threshold, any bound repressor is “knocked-off” the operator leading to expression of the *lac* genes. (d) At high intracellular inducer concentrations the genes for lactose metabolism are fully induced and eventually inducer in the environment is depleted. (e) After inducer inside the cell is depleted, repressor rebinds to the operator preventing further expression of the *lac* operon and the enzymes for lactose metabolism are either degraded or diluted through cellular division.

to) data published in a series of single-molecule *in vivo* studies regarding the *lac* systems from the Xie laboratory [258; 276; 291]. Using parameters from single-molecule experiments provided two advantages: first, the parameters were measured under *in vivo* conditions; and second, the data were not from ensemble measurements, in which the measured mean value may not correspond to any individual cell.

### 6.2 Stochastic Models of the *lac* Circuit

#### 6.2.1 Well-stirred and spatially resolved models

In order to determine the effect of spatial degrees of freedom of the *lac* circuit, we needed two models of the system for comparison: a well-stirred model and a spatially resolved model. Both of the models were
stochastic, with the well-stirred model taking a CME approach and the spatial model a reaction-diffusion master equation (RDME) approach. The RDME method can be thought of as a superset of the CME in that all of the kinetic rates used for modeling reactions in the CME based model are also used in the RDME model, but with additional parameters regarding the spatial localization of particles and their diffusion in three-dimensional space.

Additionally, our spatial model includes an approximated cellular environment. The cellular volume is constrained similarly to a standard *E. coli* cell – a cylinder 2 μm long by 0.8 μm in diameter with spherical endcaps. The volume is surrounded by a (generally) impermeable membrane separating the extracellular environment from the intracellular. The intracellular environment is crowded with an approximated *in vivo* environment, described previously [270], and the macromolecules involved in the kinetic reactions are placed either in specific locations or randomly distributed in the intracellular space.

Since both stochastic models are more complex than can be solved using analytic methods, we used computational Monte Carlo methods to sample the master equations and determine the probability distributions. The well-stirred model was sampled using a standard Gillespie technique and the spatial model was sampled using our lattice microbe method, as presented earlier.

<table>
<thead>
<tr>
<th>Reaction</th>
<th>Param</th>
<th>TMG</th>
<th>IPTG</th>
<th>Units</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R_2 + O \rightarrow R_2O$</td>
<td>$k_{tron}$</td>
<td>2.43e+06</td>
<td></td>
<td>M$^{-1}$s$^{-1}$</td>
<td></td>
</tr>
<tr>
<td>$R_2O \rightarrow R_2 + O$</td>
<td>$k_{tioff}$</td>
<td>6.30e-04</td>
<td></td>
<td>s$^{-1}$</td>
<td></td>
</tr>
<tr>
<td>$IR_2 + O \rightarrow IR_2O$</td>
<td>$k_{iron}$</td>
<td>1.21e+06</td>
<td></td>
<td>M$^{-1}$s$^{-1}$</td>
<td></td>
</tr>
<tr>
<td>$IR_2O \rightarrow IR_2 + O$</td>
<td>$k_{iioff}$</td>
<td>6.30e-04</td>
<td></td>
<td>s$^{-1}$</td>
<td></td>
</tr>
<tr>
<td>$I_2R_2 + O \rightarrow I_2R_2O$</td>
<td>$k_{2tron}$</td>
<td>2.43e+04</td>
<td></td>
<td>M$^{-1}$s$^{-1}$</td>
<td></td>
</tr>
<tr>
<td>$I_2R_2O \rightarrow I_2R_2 + O$</td>
<td>$k_{2iioff}$</td>
<td>3.15e-01</td>
<td></td>
<td>s$^{-1}$</td>
<td></td>
</tr>
<tr>
<td>$O \rightarrow O + mY$</td>
<td>$k_{tr}$</td>
<td>1.26e-01</td>
<td></td>
<td>s$^{-1}$</td>
<td></td>
</tr>
<tr>
<td>$mY \rightarrow mY + LacY$</td>
<td>$k_{tn}$</td>
<td>4.44e-02</td>
<td></td>
<td>s$^{-1}$</td>
<td>[276]</td>
</tr>
<tr>
<td>$mY \rightarrow \emptyset$</td>
<td>$k_{degm}$</td>
<td>1.11e-02</td>
<td></td>
<td>s$^{-1}$</td>
<td>[276]</td>
</tr>
<tr>
<td>$LacY \rightarrow \emptyset$</td>
<td>$k_{degm}$</td>
<td>2.10e-04</td>
<td></td>
<td>s$^{-1}$</td>
<td></td>
</tr>
<tr>
<td>$I + R_2 \rightarrow IR_2$</td>
<td>$k_{ion}$</td>
<td>2.27e+04</td>
<td>9.71e+04</td>
<td>M$^{-1}$s$^{-1}$</td>
<td>$C_{50} = 17.6 \mu M, 4.12 \mu M, [292; 293]$</td>
</tr>
<tr>
<td>$IR_2 \rightarrow I + R_2$</td>
<td>$k_{iioff}$</td>
<td>2.00e-01</td>
<td></td>
<td>s$^{-1}$</td>
<td></td>
</tr>
<tr>
<td>$I + IR_2 \rightarrow I_2R_2$</td>
<td>$k_{2ion}$</td>
<td>1.14e+04</td>
<td>4.85e+04</td>
<td>M$^{-1}$s$^{-1}$</td>
<td></td>
</tr>
<tr>
<td>$I_2R_2 \rightarrow I + IR_2$</td>
<td>$k_{2iioff}$</td>
<td>4.00e-01</td>
<td></td>
<td>s$^{-1}$</td>
<td></td>
</tr>
<tr>
<td>$I + R_2O \rightarrow IR_2O$</td>
<td>$k_{opon}$</td>
<td>6.67e+02</td>
<td>2.24e+04</td>
<td>M$^{-1}$s$^{-1}$</td>
<td>$C_{50-op} = 3 \mu M, 89.4 \mu M, [292; 293]$</td>
</tr>
<tr>
<td>$IR_2O \rightarrow I + R_2O$</td>
<td>$k_{opoff}$</td>
<td>1.00e+00</td>
<td></td>
<td>s$^{-1}$</td>
<td></td>
</tr>
<tr>
<td>$I + IR_2O \rightarrow I_2R_2O$</td>
<td>$k_{2opon}$</td>
<td>3.33e+02</td>
<td>1.12e+04</td>
<td>M$^{-1}$s$^{-1}$</td>
<td></td>
</tr>
<tr>
<td>$I_2R_2O \rightarrow I + IR_2O$</td>
<td>$k_{2opoff}$</td>
<td>2.00e+00</td>
<td></td>
<td>s$^{-1}$</td>
<td></td>
</tr>
<tr>
<td>$Iex \rightarrow I$</td>
<td>$k_{id}$</td>
<td>1.53e-02</td>
<td></td>
<td>s$^{-1}$</td>
<td>[294]</td>
</tr>
<tr>
<td>$I \rightarrow Iex$</td>
<td>$k_{id}$</td>
<td>1.53e-02</td>
<td></td>
<td>s$^{-1}$</td>
<td></td>
</tr>
<tr>
<td>$LacY + Iex \rightarrow LacYI$</td>
<td>$k_{yion}$</td>
<td>2.00e+05</td>
<td></td>
<td>M$^{-1}$s$^{-1}$</td>
<td></td>
</tr>
<tr>
<td>$LacYI \rightarrow LacY + Iex$</td>
<td>$k_{yioff}$</td>
<td>1.00e+01</td>
<td></td>
<td>s$^{-1}$</td>
<td></td>
</tr>
<tr>
<td>$LacYI \rightarrow LacY + I$</td>
<td>$k_{it}$</td>
<td>1.00e+03</td>
<td></td>
<td>s$^{-1}$</td>
<td></td>
</tr>
</tbody>
</table>
6.2.2 Lac repressor/inducer kinetics

The lac repressor (LacI; R in the model annotation) [295–297] rapidly dimerizes with high affinity and the dimers further associate to form tetramers with a $K_d$ in the nanomolar range [298]. The tetrameric form enhances repression by binding multiple lac operators simultaneously. However, as the current model deals with only a single operator, we assumed a mutant form of LacI that did not tetramerize. Furthermore, we assumed that the dimerization $K_d$ was sufficiently low that LacI only existed in the dimer state, the species $R_2$. Ten molecules of $R_2$ were randomly placed in the cell and we assumed that the cell regulated this number to be constant, i.e., we assumed no noise from the transcription/translation of the repressor gene.

Inducer molecules (I) are small sugar-like solutes and can both passively diffuse and be actively transported across the cellular membrane. Inducer molecules in the extracellular space, $I_{\text{ex}}$, and those in the intracellular space, I, can diffuse across the membrane freely in both directions. Following Stamatakis et al. [294] we modeled diffusive influx and efflux as first order reactions with equal kinetic constants,

$$I_{\text{ex}} \xrightarrow{k_{\text{id}}} I.$$

Additionally, inducer molecules can be actively transported by LacY into the cell from the extracellular space. This reaction was modeled as an irreversible Michaelis-Menten reaction,

$$LacY + I_{\text{ex}} \xrightarrow{k_{\text{yion}}} LacYI \xrightarrow{k_{\text{yoff}}} LacY + I.$$

These rate constant were also taken from [294], but modified to give the appropriate 65-fold increase in intracellular concentration for the inducer thiomethyl-$\beta$-D-galactoside (TMG) and a 40-fold increase for the inducer isopropyl-$\beta$-D-1-thiogalactopyranoside (IPTG) in the fully induced state.

Inducer molecules in the extracellular space are maintained at a constant concentration. In the Gillespie simulations this is achieved by creating another $I_{\text{ex}}$ molecule whenever one moves from outside to inside the cell and destroying an $I_{\text{ex}}$ molecule whenever one moves from inside to outside. In the lattice microbe simulations the lattice is connected to a infinite reservoir of $I_{\text{ex}}$ molecules through the use of constant concentration boundary conditions.

In the lattice microbe model, the small inducer molecules diffuse at either 1000 $\mu$m$^2$/s in extracellular space and 100 $\mu$m$^2$/s in intracellular space or at the maximum lattice diffusion coefficient, whichever is slower. Since the maximum diffusion coefficient depends on the lattice spacing and time step, for certain simulations it was necessary to limit the value to lower than measured to reach simulation times on the order of the cell cycle. Since inducer molecules are present in large numbers and they always diffuse faster than proteins this approximation is not expected to have a noticeable effect.

Upon entering the cell, inducer molecules bind to free LacI. Each LacI monomer can bind a single
inducer molecule hence an R₂ dimer can bind two inducers:

\[ I + R₂ \xrightarrow{k_{ion}} I_2 R₂, \]  
\[ I + IR₂ \xrightarrow{k_{2on}} IR₂, \]  
\[ I + IR₂ \xrightarrow{k_{2off}} I_2 R₂. \]  

In vitro kinetic data suggests non-cooperative binding (Hill coefficient of 1) of inducer to R₂ in the absence of lac operator DNA [292; 299; 300], corresponding to \( k_{2on} = \frac{1}{2} k_{ion} \) and \( k_{2off} = 2 \cdot k_{off} \). To see this relationship, consider the following mass balance equations:

\[ \frac{d[R₂]}{dt} = -k_{ion}[I][R₂] + k_{off}[IR₂], \]  
\[ \frac{d[IR₂]}{dt} = k_{ion}[I][R₂] - k_{off}[IR₂] - k_{2on}[I][IR₂] + k_{2off}[I₂ R₂], \]  
\[ \frac{d[I₂ R₂]}{dt} = k_{2on}[I][IR₂] - k_{2off}[I₂ R₂]. \]

Solving for the steady state, by setting these equations equal zero, and using the expression for the total number of repressor dimers \( R₂T = [R₂] + [IR₂] + [I₂ R₂] \), it can be shown that the fraction of LacI monomers bound to an inducer molecule is given by

\[ f_B = \frac{(2[I] + K_{d(i2)})[I]}{2([I]^2 + K_{d(i2)}[I] + K_{d(i)} \cdot K_{d(i2)})}, \]  

where the equilibrium dissociation constants \( K_{d(i)} = \frac{k_{off}}{k_{ion}} \) and \( K_{d(i2)} = \frac{k_{2off}}{k_{2on}} \) have been used. The concentration of inducer resulting in half of the repressors monomers being bound is then

\[ C_{50} = \sqrt{K_{d(i)} \cdot K_{d(i2)}}. \]  

Cooperativity can be described using the Hill equation, \( f_B = \frac{[L]^n}{(C_{50})^n + [L]^n} \), where the Hill coefficient \( n \) is indicative of the degree of cooperation between multiple ligands \( L \) binding to a macromolecule. Formulating inducer binding as a Hill equation with coefficient of 1 gives

\[ f_B = \frac{[I]}{\sqrt{K_{d(i)} \cdot K_{d(i2)}} + [I]}. \]  

Equating Equations 6.5 & 6.7 it becomes apparent that Equations 6.1a & 6.1b are non-cooperative only when \( K_{d(i2)} = 4 \cdot K_{d(i)} \). If the probability of inducer unbinding is independent of the number of inducers bound, the rate of unbinding when two inducers are bound will be twice that when a single inducer is bound: \( k_{2off} = 2 \cdot k_{off} \). Therefore, \( k_{2on} = \frac{k_{2off}}{K_{d(i2)}} = \frac{2 \cdot k_{off}}{4 \cdot K_{d(i)}} = \frac{1}{2} k_{ion} \).

Inducer molecules also bind to the repressor-operator complex, but with a much lower affinity. Although there is some equilibrium data suggesting this binding is cooperative with a Hill coefficient of 1.45 [292], such cooperativity was not observed in kinetic measurements of binding and unbinding [293]. For simplicity,
we therefore assumed a non-cooperative model

\[ I + R_2O \xrightarrow{k_{\text{ippon}}} IR_2O, \]  
\[ I + IR_2O \xrightarrow{k_{\text{ipoff}}} I_2R_2O, \]  

(6.8a)  
(6.8b)

with \( k_{i2\text{opon}} = \frac{1}{2}k_{\text{ipon}} \) and \( k_{i2\text{opoff}} = 2 \cdot k_{\text{ipoff}} \).

We wanted to investigate the behavior of the lac circuit using both of the inducers IPTG and TMG, which differ in their C50 value. Kinetic rate constants [292] and equilibrium binding measurements [293] were available for IPTG binding to both free repressor and the repressor-operator complex. From the kinetic measurements, the rate constants for inducer binding and unbinding were \(4.85 \cdot 10^4 \text{M}^{-1} \text{s}^{-1}\) and \(0.2 \text{ s}^{-1}\) for the free repressor and \(1.12 \cdot 10^4 \text{M}^{-1} \text{s}^{-1}\) and \(1.0 \text{ s}^{-1}\) for the repressor-operator complex. This yields a C50 for binding to the repressor-operator complex (89 µM) that is \(\sim 20\) times higher than for free repressor (4.1 µM). The stochastic rate constants corresponding to these values are given in Table 6.1. Figure 6.2 shows the results of using these rate constants in stochastic simulations of inducer binding; good agreement between simulations and experiments were seen for both kinetic and equilibrium measurements. TMG has been reported to have a C50 for binding to free repressor greater than that for IPTG by a factor of \(\sim 10\) [301]. However, since neither kinetic data nor detailed equilibrium studies were available, we assumed the same unbinding rate constants for TMG as IPTG and fit the rate constants for binding to both free repressor and repressor-operator complex to single molecule in vivo measurements. The details of this procedure are presented in Section 6.3.
6.2.3 Lac operon regulation

The regulatory behavior of the lac circuit results from the binding of the repressor to the lac operator, thereby inhibiting transcription initiation. There were three possible repressor species in the model, and we included in the model the binding and unbinding of each species to the operator:

\[
\begin{align*}
R_2 + O & \overset{k_{ron}}{\underset{k_{roff}}{\rightleftharpoons}} R_2O, \\
IR_2 + O & \overset{k_{iron}}{\underset{k_{iroff}}{\rightleftharpoons}} IR_2O, \\
I_2R_2 + O & \overset{k_{i2ron}}{\underset{k_{i2roff}}{\rightleftharpoons}} I_2R_2O.
\end{align*}
\]

(6.9)

(6.10)

(6.11)

The stoichiometry of inducer–repressor binding is currently subject to debate [300]; it is unclear whether the affinity of IR2 for the operator is of the same order as that of R2. We therefore compared the effect on our model of both a high \( k_{iron} \) (comparable to \( k_{ron} \)) and a low \( k_{iron} (\ll k_{ron}) \). In either case, the affinity of \( I_2R_2 \) for the operator is thought to be low and we assumed \( k_{i2ron} \ll k_{ron} \) and \( k_{i2roff} \gg k_{roff} \). Values for the rate constants were obtained by fitting the model with experimental LacY distributions from single cells, as presented in Section 6.3.

6.2.4 Transcription, translation, and degradation

Transcription initiation of a LacY mRNA (mY) from the lac operon was modeled as a first order process dependent on a free O operator, using a mean field approximation for RNA polymerase (RNAP) complexes:

\[ O \overset{k_{tr}}{\rightarrow} O + mY. \]

(6.12)

The transcription rate constant \( (k_{tr}) \) was determined by fitting LacY distributions to published single molecule measurements, as described in Section 6.3. In the spatial model, an mY molecule was created instantaneously at the location of the operator following transcription initiation and then allowed to diffuse in the cytoplasm with a diffusion constant \( D_{mY} = 0.1 \mu m^2/s \) [302; 303].

Decay of and translation initiation from mY were modeled as a competition between RNase E enzymes [304] and ribosomes for an mY’s ribosomal binding site (RBS). Both RNase E and ribosomes were assumed to be in high and constant abundance relative to mY, so both reactions were treated as first order without accounting for RNase E or ribosomes explicitly. The rate of degradation of mY by RNase E was chosen to result in a mean lifetime \( (t_{mY}) \) of 90 s, as reported by Yu \textit{et al.} [276]. The rate of translation initiation was chosen to produce a mean of 4 LacY proteins over the lifetime of an average mY messenger, also as reported in [276]. Translation itself was considered an instantaneous process producing a single LacY per ribosome binding event at the same location as the mRNA:

\[
\begin{align*}
mY & \overset{k_{degm}}{\rightarrow} \emptyset, \\
mY & \overset{k_{tn}}{\rightarrow} mY + LacY,
\end{align*}
\]

(6.13)

(6.14)
with \( k_{\text{degm}} = 1/t_{\text{my}} \) and \( k_{\text{tn}} = 4.2/t_{\text{my}} \).

In *E. coli*, translation of an mRNA containing the sequence for integral membrane proteins is thought to be coupled with translocation of the resultant protein across the cytoplasmic membrane by the Sec translo-
case [305], *i.e.*, cotranslational translocation. Specifically, LacY has been observed to require the bacterial signal recognition particle (SRP) pathway for functional membrane integration [306–308]. In the spatial model, then, mY was required to diffuse to the membrane before translation could occur; Equation 6.14 was limited to membrane sites. Since ribosomes likely attach to an mRNA’s RBS while transcription is still ongoing [309], the model assumed that mY molecules were protected from degradation by RNase E until after mY reached the membrane; Equation 6.13 was also limited to membrane sites.

The loss of membrane proteins in *E. coli* is primarily from dilution as a result of cellular growth over the cell cycle [310]. Therefore, degradation of LacY was modeled as a first order reaction with a half-life corresponding to the cell doubling time (\( t_{\text{cell}} \)).

\[
\text{LacY} \xrightarrow{k_{\text{degp}}} \emptyset, \tag{6.15}
\]

where \( k_{\text{degp}} = \ln 2/t_{\text{cell}} \). The number of R repressor dimers was assumed to be actively maintained at a constant level during a simulation, with no production or degradation.

### 6.3 Obtaining Parameters from Single-Cell Distributions

#### 6.3.1 Linear relationship between transcriptional burst size and inducer concentration

In a recent *in vivo* single-molecule fluorescence study, Choi *et al.* measured the distributions of a fluorescent reporter protein under control of the *lac* operator in individual *E. coli* cells at various inducer (TMG) concentrations [258]. They performed the measurements in the absence of LacY’s positive feedback by replacing its gene with that of the membrane protein Tsr in the *lac* operon. This enabled an accurate determination of the protein distribution produced by the circuit at a given inducer concentration without any confounding non-linear effects due to enhancement of the internal inducer concentration by LacY. In the absence of DNA looping, they were able to fit their observed distributions to a gamma distribution

\[
P(x) = \frac{x^{a-1}e^{-x/b}}{\Gamma(a)b^a},
\]

where \( a \) was interpreted as the frequency of transcriptional bursts relative to the protein lifetime and \( b \) as the mean number of proteins produced per burst. They observed a relatively constant value for the burst frequency of 3–4 and a linearly increasing relationship between burst size and inducer concentration.

To understand the origin of the linear relationship between burst size and inducer concentration and to reproduce this behavior in our model, we derived an expression for the burst size as a function of kinetic parameters in the model. As long as bursts are infrequent relative to protein degradation, *i.e.* once a free operator is bound with a repressor it remains bound for a significant fraction of the cell cycle, transcriptional bursting from the *lacy* gene can be modeled as a Markov process with competition between RNAP and the various LacI species for binding to the free operator (see Figure 6.3).

Transcription initiation by RNAP was modeled as a pseudo first order process (Equation 6.12), with a rate constant of \( k_{\text{tr}} \). Repressor exists in two states with potentially significant binding affinity: \( R_2 \) and
IR$_2$. Free repressor binds with free operator according to Equation 6.9, which results in a pseudo first order rate of $[R_2] \cdot k_{ron}$. Since there is currently debate surrounding the binding affinity of the IR$_2$ state to the operator, we set this rate ($k_{iron}$) to be a fraction of the free repressor binding constant and analyzed at the effect of varying the proportionality constant $k$. The pseudo first order rate then is $[IR_2] \cdot k \cdot k_{ron}$. This model of transcriptional bursting assumes that the binding of IR$_2$ to the free operator is negligible, such that $k_{iron} \ll k_{ron}$. In practice, this condition was satisfied when $k_{iron} \leq \frac{k_{ron}}{100}$. We used the upper limit $k_{iron} = \frac{k_{ron}}{100}$ in our model, which is within the range experimentally reported [301].

Following the unbinding of a repressor from the repressor–operator complex, the probability of tran-
scription initiation (and subsequent mRNA creation) occurring at the free operator as opposed to a repressor re-binding is

$$p = \frac{k_{tr}}{k_{tr} + [R_2] \cdot k_{ron} + [IR_2] \cdot k \cdot k_{ron}}. \quad (6.16)$$

The probability of a given number of consecutive transcription initiation events (the size of the mRNA burst) then follows a geometric distribution with $P(n) = p^n(1-p)$ of which the mean is $\sum_{n=0}^{\infty} P(n) \cdot n = \frac{p}{1-p}$. However, a repressor unbinding event that produces no mRNA is not observable as a burst, the burst size (B) is therefore the mean size of all bursts producing at least one mRNA

$$B = \text{mean}(n > 0) = \frac{\sum_{n=1}^{\infty} P(n) \cdot n}{\sum_{n=1}^{\infty} P(n)},$$

$$= \frac{\sum_{n=0}^{\infty} P(n) \cdot n}{1 - P(0)}$$

$$= \frac{p}{1-p} \cdot \frac{1}{1 - (1-p)},$$

$$= \frac{1}{1-p}. \quad (6.17)$$

Combining Equations 6.16 and 6.17 gives the expression for the mean burst size in terms of the rate constants.
Figure 6.4: (left) Mean burst size as a function of inducer concentration for various values of \( k \), where \( k_{iron} = k \cdot k_{ron} \). Parameters used were \( R_{2T} = 2.08 \cdot 10^{-5} \) M, \( C_{50} = 1.76 \cdot 10^{-5} \) M, \( k_{tr} = 1.26 \cdot 10^{-1} \) s\(^{-1} \), and \( k_{ron} = 2.43 \cdot 10^{6} \) M\(^{-1}\)s\(^{-1} \). (right) The rate of change in the burst size with respect to the inducer concentration.

for transcription initiation and repressor binding

\[
B = \frac{k_{tr}}{k_{ron}([R_2] + k \cdot [IR_2])} + 1. \tag{6.18}
\]

Given the inducer mass balances from Equations 6.2–6.4 and the expression for the total number of repressor dimers \( R_{2T} = [R_2] + [IR_2] + [I_2R_2] \), one can derive the equilibrium concentrations of the two repressor species

\[
[R_2] = \frac{(C_{50})^2 \cdot R_{2T}}{(C_{50} + [I])^2},
\]

\[
[IR_2] = \frac{2 \cdot C_{50} \cdot R_{2T} \cdot [I]}{(C_{50} + [I])^2},
\]

where \( C_{50} \) is the inducer concentration at which half of the repressor monomers are bound to an inducer molecule (Equation 6.6). Substituting \([R_2]\) and \([IR_2]\) into Equation 6.18 gives the expression for the burst size as a function of inducer concentration

\[
B = \frac{k_{tr} (C_{50} + [I])^2}{k_{ron} \cdot C_{50} \cdot R_{2T} (C_{50} + 2 \cdot k \cdot [I])} + 1. \tag{6.19}
\]

From this last equation it is clear that the burst size will be linear over the entire range of inducer concentrations only when \( k = \frac{1}{2} \). Figure 6.4 shows the effect of varying \( k \), of particular interest are the very low values of \( k \). When \( k \ll 1 \), the burst size does not linearly increase over the range of inducer concentrations for which this behavior has been reported (0-200 \( \mu \) M). In the model here formulated, a linear relationship between burst size and inducer concentration exists only when the binding affinity of \( IR_2 \) for the free operator is comparable to that of \( R_2 \). For our simulations, we therefore chose \( k = \frac{1}{2} \), such that \( k_{iron} = \frac{k_{ron}}{2} \), as this value gives a strictly linear relationship for all inducer concentrations and is amenable for analysis.
6.3.2 Fitting transcription and inducer/repressor rate constants to single-cell distributions

To obtain values for the remaining rate constants in the model, we used the distributions for LacY reported by Choi et al. [258], specifically the inferred burst frequency and size parameters ($a$ and $b$) from their gamma distribution fits. From Equation 6.19, the mean burst size as a function of inducer concentration is

$$B = \frac{k_{tr}}{k_{ron} \cdot R_{2T}} \cdot \frac{[I]}{C_{50}} + 1 + \frac{k_{tr}}{k_{ron} \cdot R_{2T}}.$$

This equation is linear in inducer concentration and by fitting the equation to the experimental burst sizes, as shown in Figure 6.5, one can constrain the kinetic parameters. The y-intercept of the line fixes the ratio of transcription to repression in the uninduced state ($\frac{k_{tr}}{k_{ron} \cdot R_{2T}}$) and the slope can then be used to obtain $C_{50} = 17.6 \mu M$.

The linear fit, however, only fixes the ratio between $k_{tr}$ and $k_{ron}$. To recover unique values for these two rate constants, we next considered the mean duration of each burst. The interpretation of the shape parameter $a$ of the gamma distribution as the burst frequency is only meaningful if the burst duration is short compared to the protein lifetime [281]. In that case, individual exponentially sized bursts can be considered exponentially distributed in time and therefore add independently to give rise to a gamma distribution of protein abundance. In setting rate constants for the model, then, we wanted to ensure that the burst duration was appropriately short.

The burst duration ($B_d$) is simply the mean time for a repressor to bind to a free operator. Given a constant $k_{tr}$, a linear relationship between burst size and inducer concentration also implies a linear relationship between burst duration and inducer concentration as can be seen from

$$B_d = \frac{1}{k_{ron}([R_2] + k \cdot [IR_2]).}$$

$$= \frac{(C_{50} + [I])^2}{k_{ron} \cdot R_{2T} \cdot C_{50} + 2 \cdot k \cdot [I]},$$

$$= \frac{1 + [I]}{k_{ron} \cdot R_{2T}.}$$

(6.20)

where $k = \frac{1}{2}$ in the last step. For TMG, the linear relationship between burst size and inducer concentration extended to at least $\sim 200 \mu M$, which is $\sim 11$ times the $C_{50}$ value for TMG of $17.6 \mu M$. From Figure 6.6 it
can be seen that the interpretation of $a$ as the burst frequency begins to break down once the burst duration is $> 5\%$ of the protein lifetime. Using $5\%$ of the protein lifetime as the burst duration for 200 $\mu$M, we can compute the value for $k_{\text{ron}}$ that gives the appropriate burst duration: $k_{\text{ron}} = 12/(0.05 \cdot \frac{t_{\text{cell}}}{ln2} \cdot R_{2T}) = 2.43 \cdot 10^6$ $\text{M}^{-1}\text{s}^{-1}$, using a cell doubling time of 55 minutes. With this value for the repressor binding rate, a single repressor molecule in an E. coli cell would take $\sim 200$ s to find a free operator. This is faster than the 354 s reported by Elf et al. [291], but of the same order of magnitude and quite reasonable given the assumptions of the model.

Then, using the above value for $k_{\text{ron}}$ and the ratio of $k_{\text{tr}}$ to $k_{\text{ron}}$ from the linear fit of the experimental data we obtained the value for the transcription rate $k_{\text{tr}} = 1.26 \cdot 10^{-1}$ $\text{s}^{-1}$. This rate for transcription initiation results in a steady state concentration of $2000$ LacY molecules per cell in the fully induced state, within a factor of two of the $\sim 1000-1200$ reported in the literature [258; 311].

In order to reproduce a burst frequency of $a$ over the LacY lifetime in the model, the repressor should dissociate from the operator with a frequency $k_{\text{roff}} = a \cdot ln2 / t_{\text{cell}}$, assuming that each dissociation event produces a burst and that the burst duration $\ll$ the cell cycle. The burst frequencies inferred by Choi et al. for TMG levels $\leq 100$ $\mu$M are relatively constant with a mean of $\sim 3$ bursts. This corresponds to $k_{\text{roff}} = 6.30 \cdot 10^{-4}$ $\text{s}^{-1}$. Since the dissociation of a repressor dimer is not thought to be significantly affected by the binding of a single inducer molecule, $k_{\text{i2roff}} = k_{\text{roff}}$. The affinity of a repressor dimer with two bound inducer molecules, however, is thought to be much lower, i.e., the binding of a second inducer molecule essentially knocks the repressor off of the operator. In the absence of this effect, the response to an increase in inducer concentrations would take a significant fraction of the cell cycle. Elf et al. reported a response time of $< 60$ seconds for addition of IPTG to concentrations from 50 $\mu$M – 1 mM [291]. Therefore, we fit $k_{\text{i2roff}}$ such that the response of the model to increase in IPTG agreed with the published data. The best fit value was obtained for $k_{\text{i2roff}} = 500 \cdot k_{\text{roff}}$ (shown in Figure 6.7 (a)).
The final kinetic rates to be defined are those regarding the binding of TMG to the repressor–operator complex (Equations 6.8a & 6.8b). As discussed in Methods, we used the same dissociation rates as for IPTG, leaving only the association rates $k_{i2op}$ and $k_{i2pon}$, both of which can be derived from the $C_{50-op}$ value for binding of inducer to the repressor–operator complex. Figure 6.7 (b) shows the effect of varying $C_{50-op}$ on the burst frequency. As $C_{50-op}$ approaches $C_{50}$, the burst frequency begins to diverge from its expected value. This is due to the increasing occupancy of the $I_2R_2O$ state, which can decay much more quickly into a free operator than the other repressed states; with operator free more often, there are more bursts over the lifetime of a protein. A value of 3 mM for $C_{50-op}$ gave the best agreement with the experimental burst frequencies for TMG.

6.4 Effects of Spatial Heterogeneity and In Vivo Crowding

6.4.1 Well-stirred population distributions

Using the derived rates, we performed well-stirred simulations of the model in the absence of LacY positive feedback to obtain the stationary state LacY distributions as a function of internal inducer concentration, as shown in Figure 6.8(a)-(d). These distribution agree well with those reported by Choi et al. and show wide cell-to-cell variability due to the intrinsic noise of the system at intermediate inducer concentrations. At high inducer concentrations the population migrates toward a less variable distribution, as expected. We also compared the actual burst frequencies and sizes from the simulations with $a$ and $b$ parameters from gamma distribution fits of the populations. As can be seen from Figure 6.8(e) and (f) the gamma distributions fit well up to $\sim100$ µM, but beyond that inducer concentration the agreement begins to break down. The discrepancy at concentrations $>100$ µM is caused by two primary factors: the burst duration and the action of inducer knocking repressor off of the operator. Increasing the repressor binding rate would improve the fit by decreasing the duration of each burst, but would cause a large increase in the total number of repressor molecules in the fully induced state, which is not supported experimentally. Alternatively, one could increase the $C_{50-op}$ value, causing less inducer instigated dissociation of the repressor–operator

![Figure 6.7: Parameter fitting for inducer–repressor–operator interactions. (a) Fraction of operator regions bound by a repressor as a function of time following an increase of IPTG to the indicated concentration. In these simulations, $k_{i2roff} = 500 \cdot k_{roff}$. (b) Number of bursts over the mean protein lifetime as a function of inducer concentration for a variety of values of the $C_{50-op}$ parameter. x are data from Choi et al. [258].](image-url)
complex, but this would decrease the responsiveness of the circuit to addition of inducer, which is also not supported experimentally. Clearly, in order for the model to have greater predictive power, additional features are necessary. For example, adding a delay between production of mRNA to account for the steps of RNAP open complex formation would lower the overall LacY level in the fully induced state. But lacking the in vivo details of such a process, we choose to ignore these effects and proceed with analysis of the model as is.

6.4.2 Differences in burst duration due to in vivo crowding

Having established the well-stirred stationary distribution, we next wanted to evaluate the effect of spatial degrees of freedom on the distributions. One obvious reaction subject to spatial effects is the rebinding of the repressor to the operator following an unbinding event. Immediately after unbinding, a repressor is necessarily localized near the operator, i.e. it has a memory of its location. As was shown by van Zon \textit{et al.} [261], this memory effect increases the probability of repressor rebinding at very short times compared to a well-stirred approximation. However, previous studies only studied the effect of normal diffusion following unbinding. We wanted to measure any additional effect caused by anomalous diffusion due to in vivo crowding.
Figure 6.9: The effect of *in vivo* crowding on repressor rebinding. Each line represents the mean of 5000 trajectories. (a) The observed diffusion coefficient, $D = \langle r^2 \rangle / 6t$, as a function of time scale for a repressor diffusing in a volume with the indicated fraction occupied by *in vivo* obstacles. (b) $\alpha$–exponent arising from fitting $\langle r^2 \rangle$ to a model of anomalous diffusion, $\langle r^2 \rangle = 6Dt^\alpha$. (c) The probability for a repressor to rebind with the operator before diffusing into the bulk (64 nm from operator) following unbinding, as a function of the *in vivo* packing. (d) The distribution of escape times for repressors that diffuse to bulk rather than rebind, at three packing values.

To investigate repressor rebinding in an *in vivo* environment, we performed reaction-diffusion simulations of a 128 nm × 128 nm × 128 nm volume centered on an operator immediately following unbinding of a repressor. We varied the packing density of the approximated *in vivo* environment to study its effect on rebinding. Figure 6.9(a) and (b) shows that there is indeed an anomalous effect at short time scales (<1 ms). Repressor diffusion at very short time scales is normal at the *in vitro* rate, diffusion between 1–100 µs is the period of maximal anomalous behavior, and diffusion at very long time scales returns to normal diffusion with a smaller D. Brownian dynamics simulations of a virtual *in vivo* environment show a similar effect with a minimum in the $\alpha$ parameter of ~0.9 for proteins close in size to a repressor dimer (75 kDa) [312]. The closest *in vivo* packing fraction corresponds to 50% obstructed by volume.

The anomalous behavior of the repressor causes it to spend more time near operator following unbinding than would be expected for a purely diffusive process, leading to more encounters with the operator and a
potentially greater probability of rebinding. To measure the change in rebinding probability, we counted the number of repressors that rebound to the operator following unbinding versus the number that escaped into bulk solution, defined here as leaving the simulation volume. As can be seen in Figure 6.9(c), as the density of \textit{in vivo} crowding increases, the probability of rebinding goes up. Compared to an \textit{in vitro} unpacked environment at 15\% probability of rebinding, at 50\% packing the probability of rebinding is 24\%. The distribution of escape times also broadens (Figure 6.9(d)) with particles in general taking longer to diffuse away. The anomalous effect cause the duration of some bursts to be significantly shorter than expected.

To study the effect of burst duration differences on the stationary LacY distributions in a population, we used our lattice microbe method to generate trajectories of spatially resolved \textit{E. coli} cells (see Section 6.5). Beginning with the stationary distribution from the well-stirred population, 200 cells were simulated at four internal inducer concentrations for the duration of a cell cycle (55 minutes). Over the course of the simulations, distributions in the \textit{in vivo} models gradually migrated to lower mean values and lower noise, as can be seen in Figure 6.10. Two factors caused this migration: first, the lower burst durations described above result in fewer proteins being produced and, second, the effective increase in repressor due to the decreased reaction volume. In contrast to spatial effects in an \textit{in vitro} environment, it appears that \textit{in vivo} crowding can affect both the mean value and the noise in distributions of observables. Since bacterial cells such as \textit{E. coli} are known to have packing density changes during different potions of the cell cycle, this presents the possibility of measuring these \textit{in vivo} effects on living cells if the observable distributions can be accurately quantified as a function of the cell cycle.

### 6.5 Cell-scale simulation methods

In order to extend the simulation times of our lattice microbe model of \textit{E. coli} to \sim 1 hour we decreased the resolution of the simulations by increasing the time step to 50 \textmu s and increasing the lattice spacing to 16 nm. At this time step diffusion of inducer molecules was \sim 100 times slower than expected \textit{in vivo}. Since inducer is so abundant, however, we do not expect any significant error from this approximation. At a lattice
6.6 Hypothesis Testing

In addition to using data from single-molecule experiments, lattice microbe models can also be used to construct hypotheses that can be tested with single-molecule experiments. For example, LacY is known to be cotranslationally translocated, i.e., it is inserted into the cytoplasmic membrane as it is translated by a ribosome [307; 313]. It is presently unclear if the mRNA is fully transcribed before transport to the membrane for translation or if the nucleoid is organized such that the DNA transcription site is already near the membrane so that transcription and translation occur in a coupled manner (see Figure 6.11a). If mRNA transcription occurs in the cytoplasm followed by diffusion to the membrane, one would expect that following a burst of more than one messenger each would diffuse to a different location on the membrane. From there, LacY proteins would be produced in a spatially localized region (assuming the mRNA/polysome complex diffuses very slowly). If the LacY proteins could be tracked at sufficient temporal and spatial
resolution such that they do not diffuse too far before being detected, the position of translation (and of 
the mRNA) could be determined. Multiple independent origins would indicate transcription away from the 
membrane (and would also reveal directly the transcription burst size). The results of our spatial modeling 
can be used to predict the resolution limits necessary to perform such an experiment. Fig. 6.11b and c show 
that a temporal resolution of 10 ms and a spatial resolution of 50 nm would be required to infer the origin of 
a burst of LacY translation events.
Bibliography


Appendix A

Lattice-Kinetic File Format

File alignment

Each file section must start on a 16-byte (128-bit) alignment boundary in the physical file. If a section is of variable length, it must be padded during file writing to maintain the proper alignment of the next section. Padding characters should be NULL bytes (0x00). Additionally, (u)int32 fields must start on a 4-byte (32-bit) alignment boundary and (u)int64 fields must start on an 8-byte (64-bit) alignment boundary.

File layout

All sizes are given in 8-bit bytes.

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Size</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>File Magic</td>
<td>char</td>
<td>4</td>
<td>“LKCL”</td>
</tr>
<tr>
<td>Endian Check</td>
<td>uint32</td>
<td>4</td>
<td>“0x04030201”</td>
</tr>
<tr>
<td>Version</td>
<td>uint32</td>
<td>4</td>
<td>“1”</td>
</tr>
<tr>
<td>Unused</td>
<td>byte</td>
<td>4</td>
<td>–</td>
</tr>
<tr>
<td>X Size</td>
<td>uint64</td>
<td>8</td>
<td>The number of sites along the x axis.</td>
</tr>
<tr>
<td>Y Size</td>
<td>uint64</td>
<td>8</td>
<td>The number of sites along the y axis.</td>
</tr>
<tr>
<td>Z Size</td>
<td>uint64</td>
<td>8</td>
<td>The number of sites along the z axis.</td>
</tr>
<tr>
<td>Site Distance</td>
<td>uint64</td>
<td>8</td>
<td>The distance between neighboring sites along an axis in nm.</td>
</tr>
<tr>
<td>TOC Position</td>
<td>fp64</td>
<td>8</td>
<td>The location of the file table of contents in the file.</td>
</tr>
<tr>
<td>First Frame Position</td>
<td>fp64</td>
<td>8</td>
<td>The location of the first frame in the file.</td>
</tr>
<tr>
<td>First Lattice Position</td>
<td>fp64</td>
<td>8</td>
<td>The location of the first lattice in the file.</td>
</tr>
<tr>
<td>Unused Position</td>
<td>byte</td>
<td>40</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td></td>
<td>112</td>
<td></td>
</tr>
</tbody>
</table>

Frame particle data

A frame particle data section describes the particles present in each lattice site at a given time during the simulation. A frame section begins with a header describing the frame:
The frame header is immediately followed by the frame particle data. There are two different methods for storing the frame data, optimized for different particle densities. The format of the frame data is indicated by the “Method” field in the frame header. Method 2 stores the particle count and particles for every site in the lattice, regardless of the site’s occupancy. The site index is not explicitly stored, as it is implicit in the ordering of the sites in the file. The second method stores a site index, particle count, and particles for all sites containing at least one particle.

Since method 2 does not explicitly store lattice indices, it trades the overhead associated with storing these large bit-size values with a low bit-size count for each site. This method achieves high-bit-density for highly occupied lattices, but for lattices with low occupancy the overhead is large. The total number of bits required to store a lattice frame ($b_{tot}$) using method 2 is given by:

$$b_{tot} = l_x l_y l_z (b_{count} + \rho b_{part}),$$

where $l_x$, $l_y$, and $l_z$ are the lattice dimensions, $b_{count}$ and $b_{part}$ are the number of bits required to store a lattice site’s particle count and a particle’s type, respectively, and $\rho$ is the average particle density (average number of particles per site).

For a lattice with low occupancy, method 3 provides a better bit-density by storing only those sites with one or more particles. The size of a lattice frame using method 3 is:

$$b_{tot} = \begin{cases} 
\rho l_x l_y l_z (b_{index} + b_{count} + b_{part}), & \text{if } \rho \leq 1, \\
 l_x l_y l_z (b_{index} + b_{count} + \rho b_{part}), & \text{if } \rho > 1,
\end{cases}$$

where symbols are as above and $b_{index}$ is the number of bits required to store a site index. The switching density ($\rho_{switch}$), below which method 3 produces smaller files and above which method 2 does is therefore

$$\rho_{switch} = \frac{b_{count}}{b_{index} + b_{count}}.$$
The relative data sizes for the two methods as a function of particle density are shown in Figure A.1 for two typical sets of simulations parameters. Also shown in the figure are the switching densities. In general, for particle densities $< 10\%$, method 3 produces smaller file sizes.

Figure A.2 shows the effect of data compression on each of the two methods. For low occupancy, method 2 shows a high level of compression, indicating much redundant information. Method 3, on the other hand, shows little compression (1-2%) throughout the range of densities. Interestingly, the compressed sizes for method 2 data are significantly smaller than either the compressed or uncompressed sizes for method 3 data (compression ratios of 49-67%). The discrepancy raises two interesting points. First, the gzip and bzip2 compression methods are not be able efficiently find information redundancy in data encoded according to method 2. Second, there must exist other possible encoding schemes that could provide more efficient data storage that the two here explored. Such a method could reduce file sizes for low occupancy lattices by around 30%.

The structure of file section for storing frame data using method 2 (particle count for every site, only particles present) is given below. Sites are ordered by increasing x coordinate, then y coordinate, and finally z coordinate.
Figure A.2: Compressions ratios as a function of particle density for particle data encoded with the two methods of data storage (method 2=blue dotted; method 3=red solid).

**Frame Data (Method 2)**

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Size</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Section Magic</td>
<td>char</td>
<td>4</td>
<td>“FD02”</td>
</tr>
<tr>
<td>Unused</td>
<td>byte</td>
<td>4</td>
<td>–</td>
</tr>
<tr>
<td>Bits Per Count</td>
<td>uint8</td>
<td>1</td>
<td>The number of bits for each site’s particle count.</td>
</tr>
<tr>
<td>Bits Per Particle</td>
<td>uint8</td>
<td>1</td>
<td>The number of bits for each particle’s type.</td>
</tr>
<tr>
<td>Unused</td>
<td>byte</td>
<td>6</td>
<td>–</td>
</tr>
<tr>
<td>Particle Count</td>
<td>bit</td>
<td>var</td>
<td>The number of particles in the site.</td>
</tr>
<tr>
<td>Particle Type[n]</td>
<td>bit</td>
<td>var x n</td>
<td>An array listing the type of each particle at the site.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The number of entries in the array is given by the preceding count.</td>
</tr>
<tr>
<td>Padding</td>
<td>byte</td>
<td>var</td>
<td>Variable padding to ensure the section ends on a 16-byte boundary.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

16+var

The structure of file section for storing frame data using method 3 (particle count and particles present only for sites with at least one particle) is given below. Sites indices are calculated as $index = x + (y \cdot x_{size}) + (z \cdot x_{size} \cdot y_{size})$. 
Frame Data (Method 3)

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Size</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Section Magic</td>
<td>char</td>
<td>4</td>
<td>“FD03”</td>
</tr>
<tr>
<td>Unused</td>
<td>byte</td>
<td>4</td>
<td>–</td>
</tr>
<tr>
<td>Number Sites</td>
<td>uint64</td>
<td>8</td>
<td>The number of site entries that follow.</td>
</tr>
<tr>
<td>Bits Per Index</td>
<td>uint8</td>
<td>1</td>
<td>The number of bits for each site’s index.</td>
</tr>
<tr>
<td>Bits Per Count</td>
<td>uint8</td>
<td>1</td>
<td>The number of bits for each site’s particle count.</td>
</tr>
<tr>
<td>Bits Per Particle</td>
<td>uint8</td>
<td>1</td>
<td>The number of bits for each particle.</td>
</tr>
<tr>
<td>Unused</td>
<td>byte</td>
<td>13</td>
<td>–</td>
</tr>
<tr>
<td>Site Index</td>
<td>bit</td>
<td>var</td>
<td>The index of the site in the lattice.</td>
</tr>
<tr>
<td>Particle Count</td>
<td>bit</td>
<td>var</td>
<td>The number of particles in the site.</td>
</tr>
<tr>
<td>Particle[n]</td>
<td>bit</td>
<td>var x n</td>
<td>An array listing each particle at the site. The number of particles in the array is given by the preceding count.</td>
</tr>
<tr>
<td>Padding</td>
<td>byte</td>
<td>var</td>
<td>Variable padding to ensure the section ends on a 16-byte boundary.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>32+var</td>
</tr>
</tbody>
</table>

Lattice site data

In addition to the particle data for each frame, a lattice-kinetic file also stores data describing the lattice sites. Since these data change much less frequently during a simulation (possibly never after initialization) they are stored separately from the particle data in the file to minimize redundancy. Each unique lattice configuration must have its own entry in the file describing the lattice sites and indicating at which time in the simulation the configuration became active. The configuration is assumed to have been active until the activation time of the next lattice configuration.

Each lattice configuration must have a lattice header section, a lattice data section, and a lattice table of contents (TOC) section.

Lattice Header

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Size</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Section Magic</td>
<td>char</td>
<td>4</td>
<td>“LHED”</td>
</tr>
<tr>
<td>Unused</td>
<td>byte</td>
<td>4</td>
<td>–</td>
</tr>
<tr>
<td>Start Time</td>
<td>uint64</td>
<td>8</td>
<td>The time that this lattice became active in ns from the beginning of the simulation.</td>
</tr>
<tr>
<td>Method</td>
<td>uint32</td>
<td>4</td>
<td>The method used to record the lattice data.</td>
</tr>
<tr>
<td>Unused</td>
<td>byte</td>
<td>4</td>
<td>–</td>
</tr>
<tr>
<td>Next Lattice Position</td>
<td>fp64</td>
<td>8</td>
<td>The location of the next lattice in the file.</td>
</tr>
<tr>
<td>Lattice TOC Position</td>
<td>fp64</td>
<td>8</td>
<td>The location of the lattice table of contents in the file.</td>
</tr>
<tr>
<td>Unused</td>
<td>byte</td>
<td>72</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>112</td>
</tr>
</tbody>
</table>
Lattice Data

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Size</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Section Magic</td>
<td>char</td>
<td>4</td>
<td>“LD02”</td>
</tr>
<tr>
<td>Unused</td>
<td>byte</td>
<td>4</td>
<td>–</td>
</tr>
<tr>
<td>Bits Per Site</td>
<td>uint8</td>
<td>1</td>
<td>The number of bits for each lattice site.</td>
</tr>
<tr>
<td>Unused</td>
<td>byte</td>
<td>7</td>
<td>–</td>
</tr>
<tr>
<td>Site</td>
<td>bit</td>
<td>var</td>
<td>The type of lattice site.</td>
</tr>
<tr>
<td>Padding</td>
<td>byte</td>
<td>var</td>
<td>Variable padding to ensure the section ends on a 16-byte boundary.</td>
</tr>
</tbody>
</table>

16+var

Table of contents

A TOC is present for each of the file sections types; it consists of a series of entries describing the data found in the section. The position in the file of the TOC for a section is given in the section’s header. The entries present in a TOC are variable depending on the type of section the TOC summarizes, with certain entries being required and others optional for a given section type. These conditions are indicated in the definition for the TOC entry.

A TOC starts with a header giving the number of entries:

Table of Contents

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Size</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Section Magic</td>
<td>char</td>
<td>4</td>
<td>“TOCS”</td>
</tr>
<tr>
<td>Unused</td>
<td>byte</td>
<td>4</td>
<td>–</td>
</tr>
<tr>
<td>Number TOC Entries</td>
<td>uint64</td>
<td>8</td>
<td>The number of TOC entries that follow. Each entry obeys the format below.</td>
</tr>
</tbody>
</table>

16

Following the TOC header are a variable number of TOC entries, each with an entry header containing the type and length of the entry specific data. The entry specific data immediately follow the entry header:

Table of Contents Entry

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Size</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOC Entry ID</td>
<td>uint32</td>
<td>4</td>
<td>The identifier for the entry type.</td>
</tr>
<tr>
<td>TOC Entry Length</td>
<td>uint32</td>
<td>4</td>
<td>The length of the entry in bytes.</td>
</tr>
<tr>
<td>Unused</td>
<td>byte</td>
<td>8</td>
<td>–</td>
</tr>
<tr>
<td>TOC Entry Data</td>
<td>byte</td>
<td>var</td>
<td>The entry specific data.</td>
</tr>
</tbody>
</table>

16+var
The TOC header and the TOC entry header are each 16 bytes in length. They therefore end (and the entry specific data start) on a 16-byte boundary. To ensure 16-byte alignment of the entries, then, the length of each entry must be evenly divisible by 16 bytes. If an entry contains variable data, it must be padded to maintain the proper alignment. This padding is NOT included in the “TOC Entry Length” field, which gives only the length of the actual entry data. The following TOC entries are defined:

**TOC Entry: Frame Positions (ID 1)**
Required In: File
Optional In:
A list of the positions of the frame sections present in the file.

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Size</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number Frames</td>
<td>uint64</td>
<td>8</td>
<td>The number of frames in the file.</td>
</tr>
<tr>
<td>Frame Positions[n]</td>
<td>fp64</td>
<td>8 x n</td>
<td>The location of each of the frame headers in the file, listed sequentially.</td>
</tr>
<tr>
<td>Padding</td>
<td>byte</td>
<td>var</td>
<td>Variable padding to ensure the entry ends on a 16-byte boundary.</td>
</tr>
</tbody>
</table>

**TOC Entry: Lattice Configuration Positions (ID 2)**
Required In: File
Optional In:
A list of the positions of the lattice configurations present in the file.

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Size</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number Lattices</td>
<td>uint64</td>
<td>8</td>
<td>The number of lattices in the file.</td>
</tr>
<tr>
<td>Lattice Positions[n]</td>
<td>fp64</td>
<td>8 x n</td>
<td>The location of each of the lattice headers in the file, listed sequentially.</td>
</tr>
<tr>
<td>Padding</td>
<td>byte</td>
<td>var</td>
<td>Variable padding to ensure the entry ends on a 16-byte boundary.</td>
</tr>
</tbody>
</table>

**TOC Entry: Lattice Properties (ID 3)**
Required In: File
Optional In:
A summary of various properties of the lattice.

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Size</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max Particles Per Site</td>
<td>uint64</td>
<td>8</td>
<td>The maximum number of particles per site.</td>
</tr>
<tr>
<td>Max Particle Type</td>
<td>uint64</td>
<td>8</td>
<td>The maximum value for a particle type.</td>
</tr>
<tr>
<td>Max Site Type</td>
<td>uint64</td>
<td>8</td>
<td>The maximum value for a site type.</td>
</tr>
<tr>
<td>Unused</td>
<td>byte</td>
<td>24</td>
<td>–</td>
</tr>
</tbody>
</table>
TOC Entry: Frame Times (ID 4)
Required In: File
Optional In:
A list of the simulation time in ns for each frame in the file.

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Size</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number Times</td>
<td>uint64</td>
<td>8</td>
<td>The number of entries to follow.</td>
</tr>
<tr>
<td>Frame Times[n]</td>
<td>uint64</td>
<td>8 x n</td>
<td>The time that the frame occurred in ns from the beginning of the simulation.</td>
</tr>
<tr>
<td>Padding</td>
<td>byte</td>
<td>var</td>
<td>Variable padding to ensure the entry ends on a 16-byte boundary.</td>
</tr>
</tbody>
</table>

TOC Entry: Lattice Configuration Times (ID 5)
Required In: File
Optional In:
A list of the simulation time in ns for each lattice configuration in the file.

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Size</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number Times</td>
<td>uint64</td>
<td>8</td>
<td>The number of entries to follow.</td>
</tr>
<tr>
<td>Lattice Times[n]</td>
<td>uint64</td>
<td>8 x n</td>
<td>The time that the lattice configuration occurred in ns from the beginning of the simulation.</td>
</tr>
<tr>
<td>Padding</td>
<td>byte</td>
<td>var</td>
<td>Variable padding to ensure the entry ends on a 16-byte boundary.</td>
</tr>
</tbody>
</table>

TOC Entry: Particle Types (ID 101)
Required In: File
Optional In: Frame Particle Data
A list of each of the particles types present and a count of the number of occurrences of the particle types. If this entry is in a frame TOC, the count gives the number of occurrences in the frame. If it is in a file TOC, it gives the maximum number of occurrences in any frame.

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Size</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number Particle Types</td>
<td>uint32</td>
<td>4</td>
<td>The number of particle types.</td>
</tr>
<tr>
<td>Unused</td>
<td>byte</td>
<td>12</td>
<td>–</td>
</tr>
<tr>
<td>Particle Types[n]</td>
<td>uint64</td>
<td>8 x n</td>
<td>An array listing each particle type.</td>
</tr>
<tr>
<td>Particle Types Counts[n]</td>
<td>uint64</td>
<td>8 x n</td>
<td>An array giving the count for each particle type.</td>
</tr>
</tbody>
</table>
TOC Entry: Site Types (ID 201)
Required In: File, Lattice Site Data
Optional In:
A list of each of the site types present in the lattice and a count of the number of times each site type appears in the lattice. If this entry is in a lattice site TOC, the count gives the number of occurrences in the configuration. If it is in a file TOC, it gives the maximum number of occurrences in any configuration.

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Size</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number Site Types</td>
<td>uint32</td>
<td>4</td>
<td>The number of site types in the lattice.</td>
</tr>
<tr>
<td>Unused</td>
<td>byte</td>
<td>12</td>
<td>–</td>
</tr>
<tr>
<td>Site Types[n]</td>
<td>uint64</td>
<td>8 x n</td>
<td>An array listing each site type.</td>
</tr>
<tr>
<td>Site Types Counts[n]</td>
<td>uint64</td>
<td>8 x n</td>
<td>An array giving the count for each site type.</td>
</tr>
</tbody>
</table>
Appendix B

Lattice-Kinetic Simulation Parameter Tables
Figure B.1: Maximum particle concentration by lattice spacing for a 32x32x32 lattice with the specified maximum number of particles per site ($n_{max}$).
Figure B.2: Maximum particle concentration by lattice spacing for a 64x64x64 lattice with the specified maximum number of particles per site ($n_{max}$).
Figure B.3: Maximum particle concentration by lattice spacing for a 128x128x128 lattice with the specified maximum number of particles per site ($n_{max}$).
Figure B.4: Maximum particle concentration by lattice spacing for a 256x256x256 lattice with the specified maximum number of particles per site ($n_{max}$).
Figure B.5: Valid range of first order kinetic rate constants by simulation timestep.
Figure B.6: Valid range of second order kinetic rate constants by simulation timestep for a lattice with the indicated natural diffusion coefficient.
Figure B.7: Valid range of second order kinetic rate constants by lattice spacing for a lattice with the indicated natural diffusion coefficient.
Appendix C

Supporting Information

Copies of all supporting information for this dissertation including, but not limited to, source code, data files, and laboratory notes have been deposited with Zan Luthey-Schulten, UIUC. Written laboratory notebooks were physically transferred and electronic materials archived to tape and then deposited.
Curriculum Vitae

ELIJAH ROBERTS
Chemical Life Science Laboratory, A540, Mailbox 68-6
600 S Mathews Dr, Urbana, IL 61801
P:217-244-2983, F:217-244-3186
erobert3@illinois.edu
http://vidar.scs.uiuc.edu/erobert3

ACADEMIC TRAINING

March 2010  University of Illinois at Urbana-Champaign, Urbana, IL
Ph.D., Biophysics and Computational Biology
Thesis: Computational Investigations of Evolutionary Transitions during Development of the Cellular Translation and Transcription Machinery
Advisor: Zaida (Zan) Luthey-Schulten

1996  Ohio University, Athens, OH
Russ College of Engineering and Technology
B.S., Computer Science

RESEARCH EXPERIENCE

Dec 2004 – present  Graduate Research Assistant, University of Illinois
Zan Luthey-Schulten Group

○ Investigating the effect of spatial localization and molecular crowding on cellular biochemical pathways.
○ Developing software methods to utilize GPU processors to simulate in vivo reaction-diffusion models.
○ Studying the evolutionary history of the ribosome, ribosomal proteins, and other translation machinery.
○ Developing evolutionary analysis software for studying history of protein and nucleic acid sequences and structures.

PEER-REVIEWED PUBLICATIONS

* equal contributions


**INVITED TALKS, SEMINARS, AND CONFERENCE PRESENTATIONS**

- “Stochastic, cell-scale simulations under *in vivo* conditions”, PI Meeting, FIBR: From Geochemistry to the Genetic Code, Santa Fe, NM. July 13, 2009.

**OTHER PUBLICATIONS**

http://www.scs.uiuc.edu/schulten/tutorials/ribosome.

http://www.scs.uiuc.edu/schulten/tutorials/ef-tu.

http://www.scs.uiuc.edu/schulten/tutorials/evolution-classII.

**PRESS COVERAGE**


James E. Kloeppele, “Evolutionary software to be released free of charge,” UIUC News Bureau (Sep 9, 2006), http://news.illinois.edu/news/06/0918software.html

**TEACHING EXPERIENCE**

<table>
<thead>
<tr>
<th>Summer 2009</th>
<th>Teaching Assistant, NSF Center for the Physics of Living Cells Summer School, Urbana, IL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fall 2006</td>
<td>Teaching Assistant, Guest Lecturer, CHEM470 Computational Chemical Biology, UIUC</td>
</tr>
<tr>
<td>March 2006</td>
<td>Teaching Assistant, Computational Biophysics Workshop, Frankfurt, Germany</td>
</tr>
<tr>
<td>June 2005</td>
<td>Teaching Assistant, Computational Biophysics Workshop, Chicago, IL</td>
</tr>
</tbody>
</table>

**PROFESSIONAL EXPERIENCE**

<table>
<thead>
<tr>
<th>2002 – 2004</th>
<th>Fifth Third Bank, Cincinnati, OH</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Technical Lead/Manager</td>
</tr>
<tr>
<td></td>
<td>Independent Consultant</td>
</tr>
<tr>
<td>1996 – 2000</td>
<td>Fifth Third Bank, Cincinnati, OH</td>
</tr>
<tr>
<td></td>
<td>Systems Analyst</td>
</tr>
</tbody>
</table>