Stony Brook University

Synthetic Gene Design: Optimization and Analysis

A Dissertation Presented

by

Rukhsana Yeasmin

to

The Graduate School

in Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

in

Computer Science

Stony Brook University

May 2015

Stony Brook University

The Graduate School

Rukhsana Yeasmin

We, the dissertation committee for the above candidate for the Doctor of Philosophy degree, hereby recommend acceptance of this dissertation.

Steven Skiena – Dissertation Advisor
Distinguished Teaching Professor, Department of Computer Science and Engineering

I. V. Ramakrishnan – Chairperson of Defense
Professor, Department of Computer Science and Engineering

Rezaul A. Chowdhury
Assistant Professor, Department of Computer Science and Engineering

Bruce Futcher
Professor, Department of Molecular Genetics and Microbiology
Stony Brook University

This dissertation is accepted by the Graduate School.

Charles Taber
Dean of the Graduate School

Abstract of the Dissertation

Synthetic Gene Design: Optimization and Analysis

by

Rukhsana Yeasmin

Doctor of Philosophy

in

Computer Science

Stony Brook University

2015

With the advance of synthetic biology has come increased interest in designing synthetic genes which optimize protein expression. We propose new algorithms for gene design under several constraints. Our optimization criteria include finding minimum energy and maximum energy RNA structures for a given gene sequence, optimizing the amount of tRNA auto-correlation in genes and designing maximum and minimum auto-correlated sequences.

We also develop methods to analyze and interpret tiled microarray genome expression data. Statistical analysis of the viral genome expression data enables us to discover unknown facts about its life cycle, its impact on the host cell shutoff mechanism. We work on identifying novel housekeeping genes and differentially expressed genes. We further seek to cluster genes at different experimental conditions based on the expression changes across the array.

Ribosome profiling is a recently developed popular method, which gives us a global picture of the active ribosomes inside a cell. Study

of the ribosome profile data helps us interpret the overall translation mechanism and determine delays at different steps of the translation process. We analyze ribosome footprint data to predict relative residency times of ribosome (RRT) at different codons and show that RRT is correlated with the usage bias of the codons based on experimental analysis of yeast. We extend our work to predict the impact of codon-pair bias on translation process and the effect of RNA secondary structure on ribosome footprint pile-up. We also work on predicting tRNA auto-correlation effect on the translation mechanism based on the analysis results obtained from the ribosome profile data.

Contents

List of Figures

List of Tables

Acknowledgements

First and foremost, I thank Almighty for giving me the endurance, patience and perseverance to successfully complete my research work.

I would like to express my special appreciation and thanks to my advisor Professor Dr. Steven Skiena, distinguished teaching professor at Stony Brook University, who has been a tremendous mentor for me both on my research as well as on my career. He has shown the attitude and the substance of a genius. Without his supervision and constant help this dissertation would not have been possible.

My dissertation committee guided me through all these years. I would like to express the deepest appreciation to my committee chair Professor Dr. I. V. Ramakrishnan and committee members, Professor Dr. Rezaul A. Chowdhury and Professor Dr. Bruce Futcher for serving as my committee members even at hardship. I also want to thank them for letting my defense be an enjoyable moment, and for their brilliant comments and suggestions. Special thanks to Professor Dr. Bruce Futcher for his active engagement in my research work.

My sincere thanks also goes to Professor Dr. Laurie Krug, for offering me the summer internship opportunity in her group and leading me working on very exciting projects. The lesson I learnt from her was invaluable. In addition, I thank Britt Glaunsinger and her lab from the University of California at Berkeley for their active collaboration.

I wish to acknowledge Justin Gardin and Alisa Yurovsky, who played the role of valuable team members for a significant part of my research work and have been amazing help for me. I want to thank Bruce Futcher Lab, for providing me the biological data for several experiments, without which my work would be incomplete. Also thanks to Jeffrey Chen and Jesmin Jahan for their support and active involvement in part of my research work. I also thank my fellow lab-members for their support and help, special thanks goes to Yanqing Chen, Charles Ward and Rami Al-Rfou.

I thank Stony Brook University for providing me the opportunity to grow as a research scientist. I also want to thank to National Science Foundation and National Institutes of Health for their financial support granted for my

research work.

A special thanks to my family for their endless love and support. Words cannot express how grateful I am to my husband, my parents, and all other family members for all of the sacrifices they have made on my behalf. I would also like to thank all of my friends who supported me to strive towards my goal. I dedicate this dissertation to my mother, whose countless support and sacrifices sustained me thus far.

Chapter 1

Introduction

With the increase in the volume and complexity of biological data, computation has become a critical part in biological research. Sophisticated algorithmic and data analysis tools are essential for biologists to better understand complex biological systems and functions of biological molecules. Computational biology is the science of development and application of data-analytical and theoretical methods, mathematical modeling and computational simulation techniques to the study of biological, behavioral, and social systems [1].

Synthetic biology is an fundamental part of computational biology. It can be described as engineering-related approach to design new biological systems and functions not available in nature and to redesign existing biological parts and systems to carry out new functions [2]. Structure of the living organisms are determined by the genetic code they carry. Researchers work on re-designing the genetic information in the organisms to control or manipulate the features available in a particular organism on need basis, either transferring useful features from one organism to another or to control harmful features in an organism.

In this synthetic gene design process, RNA plays a critical role by interacting with other cellular components. In the last few years, novel synthetic RNA components (capable of regulating gene expression) have been designed in vivo, which sets the stage for scalable and programmable cellular behavior [3]. RNA acts as an intermediary in process of translating genetic information from DNA to protein. Information is translated in the form of triplet codes. RNA also acts as a catalyst in the cellular process and gene regulation. We work on designing RNA sequences to either obtain a targeted RNA structure or to maximize/ minimize the amount of protein production in a cell or to design vaccines.

Microarray data analysis is a popular method to investigate and analyze changes in the genomic expression. It helps researchers to assess the overall

state of a cell or organism. However the data generated from the experiments are quite large in volume and requires specialized analysis methods. We have worked on statistical analysis of tiled microarray data. We aim to study the genomic expression changes at different experimental conditions. Through careful analysis, we can determine host cell shutoff mechanisms influenced by various viral life cycles.

Ribosome profiling is a recently developed method, which enables systematic monitoring of mRNA translation processes inside the cell. It gives us a way to monitor and analyze molecular mechanism, that surpasses existing approaches in speed and accuracy. We analyze high coverage ribosome profile data, where we seek to determine the relative translation speeds of different codons. We also analyze the impact of codon-pair bias on translation mechanism. We further extend our study to correlate ribosome footprint data with the secondary structure formed by the messenger RNA and analyze the effect of tRNA auto-correlation in gene translation speed.

Our major contributions to our research work can be summarized as follows:

- *Designing RNA secondary structures in coding regions*

  Structure of the RNA is the key to its functional role. Hence predicting the correct folded RNA structure is of great value. Sophisticated regulatory structures appear within highly-constrained coding regions of genes. We study the extent to which such structures can be constructed. While predicting the secondary structures of an RNA sequence is an extensively studied problem in computational biology, the inverse problem, designing sequences based on a known structure is also important. We have worked on a particular version of the inverse RNA folding problem, where the goal is to achieve a targeted energy level. For a particular RNA structure, we designed sequences with the maximum and minimum folding energy while maintaining desired codon distribution.

- *Designing autocorrelated genes*

  Redundancy in the triplet code enables the same protein to be encoded by many different RNA sequences. Codon order in the sequence plays a major role in determining the folded RNA structure. Again, the choice of codon order is significantly important in controlling the amount of protein production, as the translation speed is largely controlled by the codon order selection. Recent studies show that the degree of tRNA auto-correlation in a coding sequence has important effects on translation speed. The tRNA pairing index (TPI) has been used widely to study the phenomenon of autocorrelation in sequences. However TPI only

counts successive transitions of tRNA usage, without regard to how far apart they occur in the sequence. We propose a new type of autocorrelation measure, DICA (Distance Incorporated Codon Auto-correlation), which weighs positional distance between codons as well as the number of transitions. We demonstrate that our DICA correlates better to the expression level of a particular gene than TPI. Finally, we devise exact and heuristic algorithms to find near optimally auto-correlated and anti auto-correlated genes for the purposes of synthetic gene design.

- *Statistical analysis of tiled microarray gene expression data*

  Microarray experiments provide genome-wide expression data. Sophisticated analysis techniques are required to correctly interpret the experiment results. We conducted experiments on *Murine Gammaherpervirus 68* (MHV68) microarray data on different experimental conditions. We performed clustering analysis on the array data to group genes based on the levels of expressions at different experimental conditions. We analyzed the data to predict differentially expressed genes and results of the analysis outcome revealed important information regarding the virus life cycle [4]. The data across the arrays were merged to find novel housekeeping genes.

- *Ribosome profile data analysis*

  Ribosome profiling is a recently established popular method which aids in studying translation mechanism and in predicting the amount of protein production inside a cell. Researchers can identify the location of translation start sites and can determine the speed and distribution of the translating ribosomes by analyzing snapshots of footprinting data [5].

  Codon usage bias refers to the differences of occurrences of synonymous codons coding for the same amino acid. Read sequences obtained from the ribosome profiling can be used to verify the effect of codon usage bias in the genome of a particular organism. It is assumed that optimal codon usage helps to achieve faster translation rates and high accuracy [6, 7, 8]. Hence, codon usage optimization is expected to be higher in highly expressed genes. As *Saccharomyces cerevisiae* is a fast-growing microorganism, ribosome footprint data analysis of this organism should reflect the optimal codon composition. Statistical analysis of the ribosome footprint data from *Saccharomyces cerevisiae* reveals that, frequent codons show lower ribosomal jam while rare codons show higher ribosome density.

Similar to but independent of codon usage bias, codon pair bias refers to the variation in the frequency of occurrences of different synonymous codon pairs. For example, in human genes the *Ala-Glu* codon pair *GCC-GAA* is strongly under-represented, even though it contains the most frequent *Ala* codon [9]. Similar to the codon usage bias, the effect of codon pair bias can also be established through the ribosome footprint data analysis.

Gene translation speed is partly regulated by the tRNA auto-correlation effect [10]. We found in *yeast*, highly expressed genes have higher auto-correlation scores [11]. However there is no direct measure of the extent to which tRNA auto-correlation contributes to the translation speed. We try to predict the effect of tRNA auto-correlation on ribosome residency time.

Ribosome profile data shows the ribosome traffic snapshots inside the cell at a particular time. If we track the number of footprints along any mRNA sequence, we can see large variation in the number of footprints generated at different positions. Several factors may contribute to the formation of read pile-ups. One potential reason behind the pile-ups can be the amount of secondary structure around a codon position. Several studies revealed that, the amount of secondary structure at the 5' end of a coding sequence plays important role in controlling the amount of protein production [8, 12, 13]. We try to correlate the number of footprints at different codon positions with the amount of secondary structures.

1.1 Overview

- We have developed novel algorithms to design RNA sequences with maximum (least stable or minimum structured) and minimum (most stable or maximum structured) folding energies. Chapter 2 describes our work on designing RNA secondary structures in coding regions.

- In chapter 3, I have demonstrated our work for designing auto-correlated genes. We have proposed a new distance incorporated measure of tRNA auto-correlation and provided a comparative study of our method with existing analysis method. Finally we have devised algorithms to design maximum and minimum auto-correlated sequences.

- Chapter 4 describes our work on the statistical analysis of tiled microarray data. Interesting observations regarding *gammaherpesvirus* life cycle came out of the analysis. We analyzed the data to predict differentially

expressed genes from a given set of genes and to cluster similar genes together. Through the analysis of the combined data-set, we worked on finding novel housekeeping genes, which can be used as control genes at later experiments.

- We developed a novel algorithmic procedure to determine the ribosome residency times of the codons. Chapter 5 includes our work on ribosome profile data analysis to measure the average decoding rates of individual codons.

- In chapter 6, our work on statistical analysis of the ribosome profile data has been described. In the previous chapter (chapter 5), role of codon order on ribosome profile data have been demonstrated. In this chapter, I have included statistical analysis of the data from different perspectives. Here, we have focused on the impact of codon-pair bias, of folded RNA secondary structure and tRNA auto-correlation effect.

Chapter 2

Designing RNA secondary structures in coding regions[1,2]

The secondary structures formed by RNA molecules are critical to understanding their molecular interactions and biological functions. Computationally predicting the secondary structure of an RNA sequence is a classical problem in computational biology, and has been extensively studied [15, 16, 17, 18]. Indeed programs such as Mfold [Mfo] and the Vienna RNA package [Vie] are widely used tools throughout molecular biology. The inverse problem, that of designing RNAs which fold into specific desired structures, is also important. RNA structures (such as transfer RNAs) are critical to a host of biological processes, motivating the need to design sequences to achieve desired shapes and functions.

We consider a particular version of the inverse RNA folding problem for gene coding sequences, where we seek to achieve a targeted energy level as opposed to a particular structure/shape. Cohen and Skiena [19, 20] have previously developed effective algorithms for designing optimal RNA sequences which code for specified amino acid sequences while maximizing or (as desired) minimizing the folding energy of the sequence. However, the gene sequences produced by these algorithms tend to use extremely skewed distributions of codons, because C-G bonds are roughly twice as stable as A-U bonds. The optimized genes thus exhibit codon usage distributions very different from that of the host organism, typically resulting in very poor expression.

Here, we study the algorithmic design of RNA sequences which code for a specific amino acid sequence using a desired distribution of codons – maximizing or minimizing the folding energy of resulting RNA. Our work is motivated

[1]R. Yeasmin and S. Skiena. Designing RNA secondary structures in coding regions. ISBRA, 2012

[2]Direct excerpts from [14]

by designing genes to modulate gene expression. Recent studies [8, 13] have shown that the amount of secondary structure on the 5' end of the coding sequence plays a critical role in maximizing protein production from a given gene. Typically synthetic genes are designed to match targeted codon distributions, but these studies suggest that RNA secondary structure should be an important part of the design considerations. Another study suggests that folding energy is an important factor in determining translation efficiency [21]. Secondary structures also play important roles as signals in viral replication, and hence designs minimizing the size of these structures have proven important in our experience. Our group routinely designs and synthesizes virus-length coding sequences for a few thousand dollars each [9, 22, 23].

Our major contributions in this work include:

- *Optimizing RNA secondary structures under codon constraints* – We present what we believe to be the first algorithms for modulating (either minimizing or maximizing) the RNA folding energy of a gene while respecting codon constraints. As described above, the demand for such tools is destined to grow as large-scale synthesis costs decline and turnaround times improve. Indeed, with our collaborators we are planning to synthesize high/low secondary structure variants of particular genes to study their effect on translation and replication.

- *Improvements in unconstrained secondary structure optimization* – We demonstrate that our algorithms produce structures with equal or less (greater for minimizing structures) energy on the codon distributions employed by the previous best inverse design algorithms. In particular, we have employed our optimization algorithms on codon distributions resulting from designs produced by the Cohen-Skiena (CS) algorithms. As validated by Mfold, we show that our algorithms design genes with better energy than those produced by [19]. This is particularly impressive as the Cohen-Skiena minimum-energy algorithm guarantees an optimal solution, albeit under a simpler energy function. These results validate the quality of our designs on wildtype codon distributions where direct comparisons for optimality are unavailable.

- *Fast estimation of folding energies following local modification* – The high $O(n^3)$ running time of traditional RNA folding algorithms limits the number of iterations possible in search-based optimization strategies like ours. We have investigated the tradeoff between the accurate but slow computations of Mfold in quickly recalculating the energy change resulting from small local changes in a given RNA sequence. We find that we generally can reduce the number of calls of this expensive operation

(and hence the running time of our algorithms) by a factor of five with little degradation in accuracy.

2.1 Preliminaries

Predicting RNA secondary structure is an important problem in computational biology. Several groups implemented algorithms for accurate energy determination of the folded RNA structure. Michael Zuker's Mfold/ UNAfold [Mfo] and Ivo Hofacker's Vienna RNA package named RNAfold [Vie] are among the most popular.

Mfold [24] uses a nearest neighbor energy rule to determine the structure. The program implements a dynamic programming (DP) technique where they maintain a DP table to store the calculated substructure energies and the optimal structure is obtained by backtracking. Dynamic programming based methods can correctly predict about 73% of known base pairs on domain of fewer than 700 nucleotides [25]. To calculate the structure energy, the entire structure is divided in parts consisting of stacked pairs, hairpins, bulges, internal loops and multi loops. There can also be single stranded bases. Total structure energy is the sum of all substructure energies. By default energy values are calculated at 37 °C. For a given RNA sequence S, Mfold program predicts the non-crossing, minimal energy structure P for S in $O(n^3)$ time and $O(n^2)$ space.

RNAfold [18] uses similar dynamic programming techniques to calculate the minimum free energy structure. The parameters used in the program are described in [25]. The Vienna RNA package uses three kinds of dynamic programming algorithms for structure prediction: the minimum free energy algorithm of Zuker and Stiegler [26] which yields a single optimal structure, the partition function algorithm of McCaskill [27] which calculates base pair probabilities in thermodynamic ensemble, and the suboptimal folding algorithm of Wuchty et.al [28] which generates all suboptimal structures within a given energy range of optimal energy. For secondary structure comparison, the package uses string alignment or tree-editing [29] methods to measure distance or dissimilarities. Finally they use inverse folding algorithm to design sequences with predefined structures, where they search for sequences folding into a predefined structure. In case of unsuccessful searches, a structure distance to the target structure is provided.

Inverse RNA Folding was first introduced in [18, 30]. RNAinverse in the Vienna RNA package [18] was developed to perform inverse RNA folding. Later an extended *Inverse RNA Folding* problem was studied by Dromi, Avihoo and Barash [31], where they added several non-structural constraints to the out-

put such as thermodynamic stability and mutational robustness. Dahiyat and Mayo [32] worked on the *Inverse Protein Folding* problem where the goal is to determine the primary sequence that folds into a given shape or structure. When they worked on it, designing the three dimensional structure from the sequence alone seemed difficult. However the inverse problem would be more tractable as one could over-engineer the system to favor the desired folding pattern.

INFO-RNA [33] is an web server for *Inverse RNA Folding* maintaining sequence constraints. They apply dynamic programming algorithm to find the initial RNA sequence that satisfies given secondary structure. It is not guaranteed to fold to the target structure as it might have another minimum free energy (mfe) structure. Thus the sequence is further processed by performing stochastic local searches to minimize the structure distance between the mfe structure of the obtained sequence and the given target structure. RNAexinv [34] is another software that performs extended *Inverse RNA folding* by considering not only the desired structure while generating the sequence but also other favorable attributes (i.e. thermodynamic stability and mutational robustness).

Stochastic context-free grammars (SCFGs) are alternative probabilistic methodologies for modeling RNA structure [35, 36]. Specific grammar rules are used to induce a joint probability distribution over all possible RNA structures and sequences. Parameters of SCFG models specify probability distributions over possible transformations that may be applied to a nonterminal symbol. These parameters do not have direct physical interpretations, they are learned from collections of RNA sequences with known secondary structures, no external laboratory experiments are needed [37].

CONTRAfold [38] is another secondary structure prediction tool which is based on a flexible probabilistic model called a conditional log-linear model (CLLM). Like SCFGs, CLLMs use the computationally driven parameter learning. However, unlike SCFGs they also have the generality to represent complex scoring schemes, such as those used in energy based predictions i.e. Mfold. CONTRAfold thus closes the gap between probabilistic and thermodynamic models.

Recently studies are being performed on improved parameter sets. Andronescu et al. [39, 40] applied Constraint-Generation and Boltzman-likelihood methods for better parameter estimation. Using these parameters they obtained much better RNA structure prediction models. Zakov et al. [41] further refined previous models by examining more types of structural elements and a larger sequential context for these elements. Their study showed that use of more detailed models with rich parameter sets improves prediction quality.

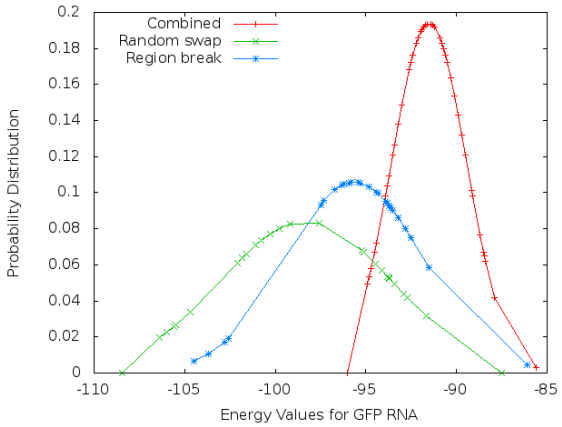

Figure 2.1: Comparison plot of min Structure Energy for *region break*, *random swap* and the combined strategy for GFP RNA of yeast. Most of the energy values for the distribution of the combined strategy (red) are higher than other two strategies.

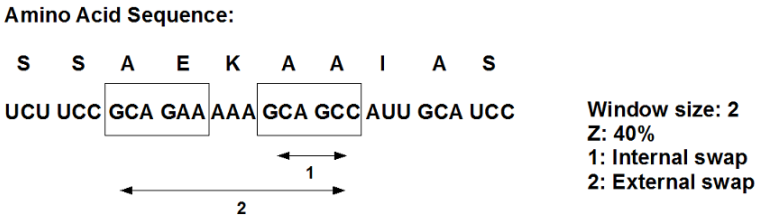

Figure 2.2: *Region break* strategy. Here, *1* shows internal swap of two codons that code for amino acid A (allow swap only in codons internal to window), *2* shows external swap of two codons corresponding to A between two windows.

Cohen and Skiena worked on *Inverse RNA folding*, where they seek the RNA sequence coding for a given protein P having minimum energy (most stable structure) over all encodings of P [19, 20]. However, as they do not pose any constraint on codon usage frequency, their designed sequences tend to use extremely skewed codon distribution. We have worked on a similar problem. However, while designing sequences we maintain the codon frequency distribution used by the host organism.

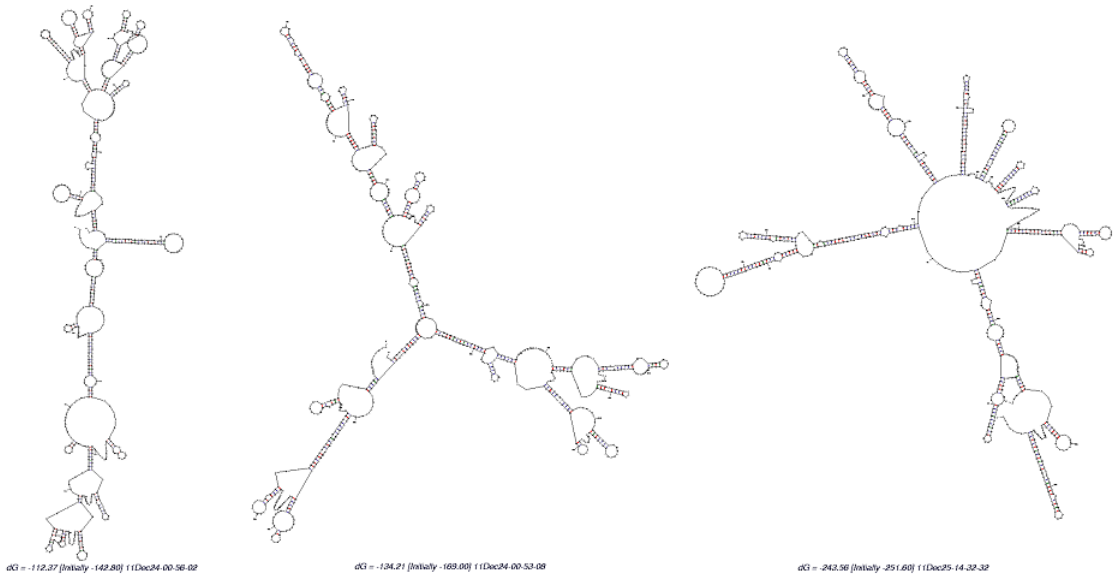

Figure 2.3: GFP RNA (jelly fish) min, wildtype and max energy folded structures. Left: folded minimum structure/ maximum energy sequence (folding energy = $-112.37 kcal/mol$). Middle: folded wildtype sequence (folding energy = $-134.21 kcal/mol$). Right: folded maximum structure/ minimum energy sequence (folding energy = $-243.56 kcal/mol$).

2.2 Heuristics for inverse RNA folding

We start with an initial wild type structure maintaining constraints imposed by the codon constraint table. The initial RNA sequence is formed by arbitrarily positioning one of the several possible codons for each amino acid. Now our goal is to find the RNA sequences with maximum and minimum energy that code for the given amino acid sequence using this codon distribution. According to Zuker and Stiegler [26], dinucleotide composition is a primary contributor to folding free energy. Our main goal was to maximize (minimize) the number of bonds to get the most (least) stable or the max (min) structured sequence. Here by most (least) stable, we mean the structure with the minimum (maximum) energy.

2.2.1 Structure maximization

To maximize structure (minimize energy) we apply *random swap* approach, where we repetitively check for codon swaps that improve the overall bond energy. We continue until we enter a state where no more improvement is possible. Here, we give the wildtype sequence to Mfold as input to find the

Algorithm 1 RNAfoldMaximizeStructure(Input: Amino Acid sequence, Codon constraints; Output: Stable structure S_1)

```
Construct initial wild type RNA sequence from the given amino acid sequence maintaining codon constraints
Call Mfold to find the current best structure S from the initial RNA sequence

while there is an improvement in the structure S do
  if no improvement in the structure energy for a specific period of time then
    Perform random changes to the structure
  else
    if the structure has previously occurred then
      Penalize the codons that have been changed in the previous step
    end if
    Find free codons for current RNA structure S
    Change the whole structure by swapping two codons that correspond to same amino acid whenever possible if the swap improves bond energy
  end if
  Call Mfold to find the best bonding S = S_1 for the modified structure
end while
```

initial bonded structure. Next we build a free codon table that keeps track of the codons that are not involved in bond formation. Hence we can replace the codons that are involved in bond formation with any other free synonymous codon listed in the free codon table, if the replacement improves the bond energy. Alternately, we can replace two free codons without breaking any existing bond.

Next, we start with a random position of the structure and check for each codon of the mRNA sequence whether swapping it with any other codon from the free codon table (which codes for the same amino acid as this one) improves the bond energy. If there is an improvement in the bond structure, we swap this codon with the free codon. We also maintain a neighbor list for each codon. Initially we allowed options, where after a swap we do not allow neighbor swap. That means, before swapping a codon we first check whether any of its neighbors has already been swapped or not. The check is performed to keep other parts of the structure relatively stable while changing one part. Later we allowed neighbors to swap and figured out allowing neighbor swap gives more freedom to find the structure with minimum energy. After checking the whole sequence, we apply the new sequence again to the Mfold program. This way we continue until we enter a state, where no more improvement is possible.

In this process sometimes we may stuck in a local minima (encounter the same sequence repeatedly). To get out of the minima, we perform some arbitrary change to the structure without violating the constraints. We penalize the codons that have been changed in the previous step so that they cannot change their position in the next step. Moreover, if there is no improvement in the structure energy for a specific number of iterations, we perform some random codon swaps to get out of local minima.

In general we run 400 iterations and the complexity in each iteration is dominated by the $O(n^3)$ running time of Mfold.

2.2.2 Structure minimization

Algorithm 2 RNAfoldMinimizeStructure(Input: Amino Acid sequence, Codon constraints; Output: Stable structure S_1)

Construct initial wild type RNA sequence from the given amino acid sequence
Call Mfold to find the current best structure S from the initial RNA sequence

while there is an improvement in the structure S **do**
  if the structure has previously occurred **then**
    Penalize the codons that have been changed in the previous step
  end if
  Apply *random swap*, or *region break*, or the combined strategy to change the current structure
  Call Mfold to find the best bonding $S = S_1$ for the modified structure
end while

To find the RNA sequence with minimum structure (maximum energy), we employed two different strategies:

- *Region break:* – Here we look for the strongest bonded parts (a fixed percentage of the total number of codons) of the current folded RNA sequence. Then we perform an internal swap of the codons corresponding to the same amino acid in these regions. For each codon, we find out the codon with maximum mismatch (based on three letters of the codon) with current codon corresponding to same amino acid and perform the swap. We repeat the process for all codons in those strongly bonded regions.

- *Random swap:* – This process is quite similar to the structure maximization process. However, here we swap two codons if the swap reduces the structure stability or maximizes energy.

We tried each of these strategies separately and also in combination. After each iteration we call Mfold for the modified structure. The combined strategy performs better than the individual ones. Fig. 2.1 shows the comparison plot of the distribution of energies for all of these strategies. From the figure, we see though *region break* strategy achieves almost similar best result as the combined strategy, the frequency of getting good results is smaller.

According to Doshi et al. [42], Mfold RNA secondary structure prediction accuracy degrades as the contact distance between base-pairs increases. One potential reason could be Mfold assumes much more long range base pairs than it occurs in general. However, in our algorithm we are maximizing (or minimizing) bonds in small local regions, so we generally avoid creating long range stem loops.

2.2.3 Algorithmic variants

We experimented with several different algorithmic variants. To find the minimum and maximum energy sequences we considered two different variations:

- *Random walk:* – Always take the current changed structure even if it is worse than the previous one.
- *Gradient descent approach:* – Before taking a bad move wait for few iterations. If after the specified number of iterations still get a worse structure, then take the best of all these bad moves as a sequence to move forward. If any of these six moves were better than the previous move but it was a duplicate one, in that case take the next good move rather than taking the duplicate one.

The *gradient descent* approach performs better than the *random walk* approach, possibly because the *random walk* approach has greater chance of allowing bad moves and trapping into local minima or maxima.

As mentioned before, for minimizing structures we used two strategies: *region break* and *random swap*. In *region break* we had to consider several different criteria. We tried with several different region sizes which was specified by window sizes (let W) and also varied the number of windows to consider at the same time. Here we used a parameter Z to specify that we will break $Z\%$ of the entire RNA sequence. Later we counted the number of windows it will take to add up to $Z\%$ of the entire RNA length. Another issue is whether

while breaking regions we should allow codons to swap only in regions internal to those windows or allow external swaps too. Initially we allow only internal swaps. If a duplicate structure is encountered, we use external swaps for the next step. Fig. 2.2 shows an example of *region break* strategy.

In *random swap* the first consideration is whether we should always start with a fixed starting point or a random one. In case of fixed starting point, we always check codons for swap from the beginning of the RNA sequence. In random starting point, all codons of the sequence starting from a randomly generated position are checked for swap in a circular manner. We found the performance of random starting point is better than fixed starting point as it allows more variations thus allowing a larger search space.The next issue is whether we should swap codons only with free codons or allow arbitrary swaps. For minimum energy structure, allowing swaps only with free codons seems reasonable as swapping two bonded codons might cause reduction in the stability of the bonded parts of the structure. For maximum energy structure it could be a reasonable one, however its performance was not good. There is a possibility that, arbitrary swap might give good result after searching for long. Here one issue is the speed of merging to a good solution. Allowing free codon swap gives good result even within a reasonably short time. Again in this case we waited five steps to get a better move before accepting a bad one.

Later for minimizing structure we combined two strategies to allow more variations while searching for the best structure. The performance of the combined one was quite better than any single one for minimizing structure as shown before in fig. 2.1. Here the question is whether we use two strategies in an interleaving manner or continue with one until we encounter a duplicate. There is no significant variation in the obtained results for any of these strategies. However, rather than changing strategy every next move, sticking with the current strategy until a duplicate structure is encountered seems reasonable as changing strategy every next move might undo the good moves of the previous step.

For structure maximization we used only the *random swap* one as breaking regions arbitrarily to find the most stable structure does not seem reasonable.

2.3 Results and discussion

It is impossible to judge the quality of a heuristic without knowledge of the correct answer. The CS program produces a provably minimum energy sequence for a model close to that of Mfold, but provides no constraint on codon usage. We evaluate our heuristics starting from a random design using the same codon distribution as optimized in the CS sequence. Thus if our program is doing a

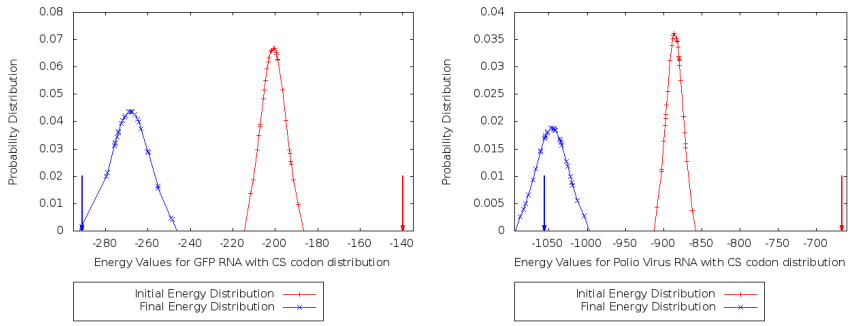

Figure 2.4: Max structure energy distribution for GFP and Polio virus RNA based on CS codon distribution; wildtype energy for $GFP = -139.8$ $kcal/mol$, for $Poliovirus = -666.6$ $kcal/mol$, CS opt energy for $GFP = -290.7$ $kcal/mol$, for $Poliovirus = -1056$ $kcal/mol$; max structure energy with CS distribution from our program is -291.3 $kcal/mol$ for GFP and -1080.6 $kcal/mol$ for Polio virus; RED arrow indicates wildtype energy and BLUE arrow indicates CS opt energy.

good job of optimization, it will produce similar energies to the optimal design. We ran our algorithm for polio virus RNA with the same codon distribution as obtained from Cohen and Skiena [19] for both minimization and maximization program, starting with a wild type sequence based on that codon distribution. Our algorithm found structures with better energy than that from CS algorithm for both minimized and maximized structures. Later we checked for GFP RNA sequence. We plotted the distribution of energy values for both max and min sequence. In fig. 2.4, the left plot shows max energy distribution for GFP RNA and the right one is for polio virus RNA based on CS codon distribution. Fig. 2.5 shows results for the same amino acid sequences maintaining given codon constraint. Fig. 2.6 and fig. 2.7 shows corresponding min structure energy distribution. Here we see, for structure minimization, our algorithm generates sequences with much higher energy than the sequences obtained from the CS program. For structure maximization, we algorithm either performs better or does as good as the CS program.

We performed experiment to compare the change in the structure energy with the change of codon distribution from wildtype to CS. Fig. 2.8 shows results from that study for both maximized and minimized structures generated randomly. Here at point 0 of X-axis, the structure follows wildtype codon distribution; at 100, it completely follows CS codon distribution. We see from fig. 2.8 that the trend of the energy plot is from lower to higher for minimized structure and from higher to lower for maximized structure as the percentage of CS codon distribution goes higher, as expected. We optimized the structures using our program. As the figure shows, we always get much better structure

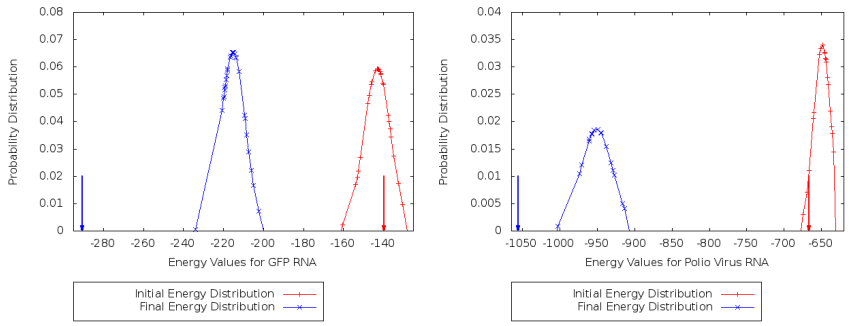

Figure 2.5: Max structure energy distribution for GFP and Polio virus RNA maintaining given codon constraints; wildtype energy for $GFP = -139.8$ $kcal/mol$, for $PolioVirus = -666.6$ $kcal/mol$, CS opt Energy for $GFP = -290.7$ $kcal/mol$, for $PolioVirus = -1056$ $kcal/mol$; max structure energy from our program is -234 $kcal/mol$ for GFP and -1003.2 $kcal/mol$ for Polio virus; RED arrow indicates wildtype energy and BLUE arrow indicates CS opt energy.

than the initial random one. However, the gap between initial energy and optimized energy decreases as we move toward CS codon distribution.

We conducted experiment on different RNA sequences (shown in table 2.1). We see from the experimental results, our algorithm always generates sequences with maximum and minimum energies compared to that of wildtype sequence energy. In most cases improvement toward minimum energy structure (max structure) is better than toward maximum energy structure (min structure). Finding the least stable structure is harder. For polio virus, our optimized max structure energy is -1003.2 $kcal/mol$ and min structure energy is -604.9 $kcal/mol$, where initial wild type energy was around -666.68 $kcal/mol$. We checked the output from CS max and min program, where they do not follow any codon constraint. The max structure energy for their program was -1056 $kcal/mol$ and min structure energy was -429.8 $kcal/mol$. We ran our program with the same codon distribution as used by CS and we obtained sequences with more optimized energies compared to the CS algorithm generated sequences (max structure energy -1080.6 $kcal/mol$, min structure energy -407.7 $kcal/mol$). This indicates our algorithm is generating the optimal sequences.

2.3.1 Parameter optimization

We have several different parameters that we need to optimize.

- For RNA structure minimization, in *region break* strategy while we are

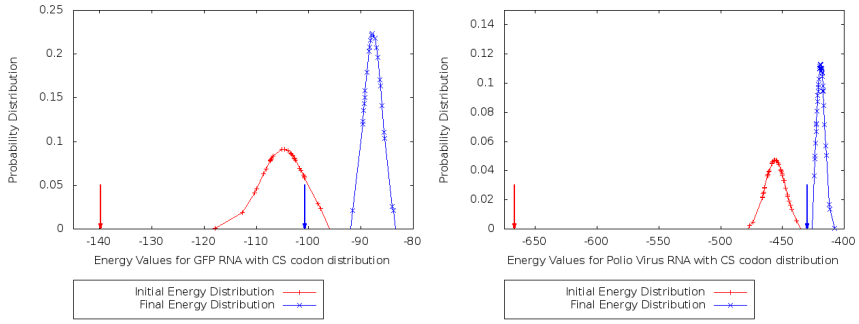

Figure 2.6: Min structure energy distribution for GFP and Polio virus RNA based on CS codon distribution; wildtype energy for $GFP = -139.8$ $kcal/mol$, for $Poliovirus = -666.6$ $kcal/mol$, CS opt energy for $GFP = -100.7$ $kcal/mol$, for $Poliovirus = -429.8$ $kcal/mol$; min structure energy with CS distribution from our program is -83.84 $kcal/mol$ for GFP and -407.7 $kcal/mol$ for Polio virus; RED arrow indicates wildtype energy and BLUE arrow indicates CS opt energy.

evaluating energy to determine the most structured parts of the folded RNA, we are moving a sliding window W around the RNA sequence, calculating energy for that window. These windows may overlap. Let P be the number of codons by which two windows overlap. Now we want to break top $Z\%$ of the entire RNA structure. We tried to determine the best values of W, P, and Z that fasten the RNA structure minimization process. We checked for $W = 10, 15, 20$ with $P = 5, 8, 10$ and for different Z values i.e. $Z = 10, 15, 20, 25$. We figured out, structure minimization process is quite independent of these parameter values. However, $Z = 15$ might be a good choice to break the structure. Fig. 2.9 shows the plot of the progress of structure minimization process for *region break* strategy with different W and P values for $Z = (10, 15)$ with the same initial wildtype RNA sequence. For other sequences the effect of these parameters are quite similar.

- Again, for structure maximization rather than calling Mfold every iteration, we reduced the number of Mfold calls which is controlled by the parameter X. For $X = i$ we call Mfold every i^{th} iteration. When $X = 1$, Mfold is called every iteration. Here, the reason to reduce the number of Mfold calls is to reduce the total running time of the algorithm, as the overall runtime is dominated by the calls to Mfold. We tried for $X = 1, 2, 3, 4, 5, 10, 20$. Experimental results show that up to $X = 5$ the algorithm's output remains the same, i.e. even running Mfold only every fifth iteration gave us the maximized structure. However, after

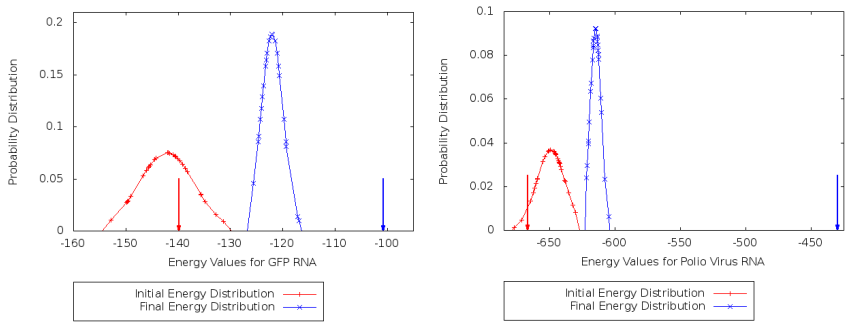

Figure 2.7: Min structure energy distribution for GFP RNA and Polio virus RNA maintaining given codon constraints; wildtype energy for $GFP = -139.8$ $kcal/mol$, for $PolioVirus = -666.6$ $kcal/mol$, CS opt energy for $GFP = -100.7$ $kcal/mol$, for $PolioVirus = -429.8$ $kcal/mol$; min structure energy from our program is -116.9 $kcal/mol$ for GFP and -604.9 $kcal/mol$ for Polio virus; RED arrow indicates wildtype energy and BLUE arrow indicates CS opt energy.

that (e.g. $X = 10, 20$) algorithm's performance degrades substantially. Fig. 2.10 shows the comparison of the distribution of energy values for different X values (left) and also the plot of the distribution of energy values for different maximum number of iterations (right). We developed an algorithm to determine energy of the current structure based on the information available in the *ct* file obtained from Mfold program. Once we know the neighbors of each codon, after swaps at each iteration we update the neighbors based on the swap. Next we calculate energy from the updated structure information. As we are updating neighbor information from the changed structure based on local swap decisions without folding the structure, it might not predict the accurate structure energy. However, we figured out, the updated information is good enough to continue changing the structure even without calling Mfold up to five iterations.

- We checked whether optimizing the same initial wild type sequence for longer is better than starting from several initial starting positions. Suppose we want to make optimum use of total time T. We want to run the algorithm in S steps with maximum iterations I at each step. Now, if each iteration of the program takes t time, then we can run $\frac{T}{t}$ iterations (say $I = i$) starting from a single step $S = 1$. Alternately, we can run several steps $S = s$ $(s > 1)$ with the number of iterations i', where $i' < i$. Here, $i \times t = s \times i' \times t$. For polio virus we found the algorithm finds the best result with approximately around 400 iterations. After that there is

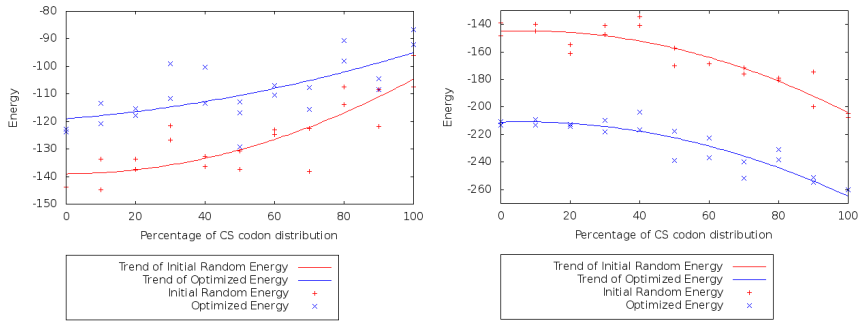

Figure 2.8: Plot of initial and optimized energy curve of GFP RNA structure for different percentage of the wildtype and CS codon distribution. Left (structure minimization): the lower curve shows trend of initial energy values; the leftmost point (0% CS codon distribution) maintains wildtype codon distribution, rightmost point maintains CS codon distribution, intermediate points maintain different ratios of codon distributions from wildtype and CS one. The upper plot shows corresponding optimized energies by RNAfoldMinimizeStructure Algorithm. Right (structure maximization): the upper curve is for wildtype structure energy and lower curve indicates the trend of energy values after maximizing structures using RNAfoldMaximizeStructure algorithm.

no improvement in the output energy. Hence, we increased the number of steps S keeping I fixed at 400 to get the optimized result. We see from the right part of fig. 2.10, the difference in energy improvement from 100 to 200 iterations is much higher than that from 200 to 400 iterations. Here, we note that the maximum number of iterations to converge to an optimal solution is dependent on the length of the RNA sequence as longer the sequence there are more options for codon swap. In general, for a sequence shorter than polio virus RNA 400 iterations should be sufficient to converge to the optimal solution. For a longer one it might take more iterations to converge.

2.4 Conclusion

In this paper, we describe programs to find the minimum and maximum energy structures of a given amino acid sequence with codon constraints. We used two algorithmic variants: *random walk* and *gradient descent* approach. Simulated annealing or the Metropolis algorithm [43] could be another approach, where at every step we could take a backward move based on a probability value. Our *gradient descent* approach is similar but we wait for several steps to find a better move before taking a bad one as Mfold is slow.

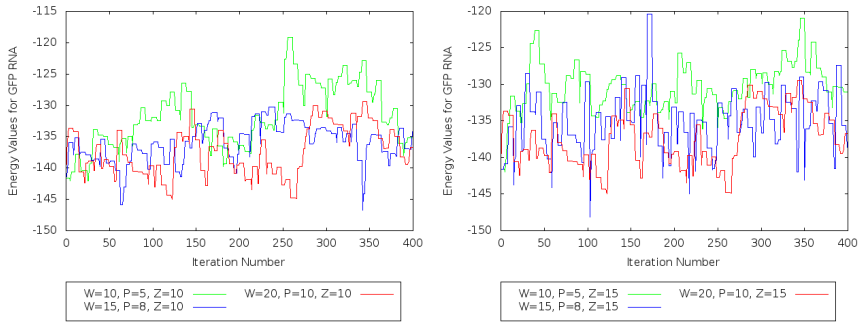

Figure 2.9: Comparison plot of *region break* strategy for different $W = 10, 15, 20$ and $P = 5, 8, 10$ with $Z = 10, 15$, where W stands for sliding window size, P is the number of overlapping codons between two windows, Z indicates the percentage of the entire structure that will be broken each step. In the figure we see, there is no significant difference in the energy optimization path that was followed for different parameter values.

We checked for several different RNA sequences. In most cases improvement toward minimum energy structure (max structure) is better than toward maximum energy structure (min structure). Finding the least stable structure is harder. For polio virus, our optimized max structure energy is -1003.2 $kcal/mol$ and min structure energy is -604.9 $kcal/mol$, where initial wild type energy was around -666.68 $kcal/mol$. We checked the output from CS max and min program, where they do not follow any codon constraint. The max structure energy for their program was -1056 $kcal/mol$ and min structure energy was -429.8 $kcal/mol$. We ran our program with the same codon distribution as used by CS and got max structure energy -1080.6 $kcal/mol$ and min structure energy -407.7 $kcal/mol$. This indicates our algorithm's performance is better than CS algorithm output, which is a clear indication of obtaining optimized structures.

2.5 Acknowledgement

Our work is available at: http://www.algorithm.cs.sunysb.edu/RNAdesign. We specially thank Yanqing Chen for assistance during the study. This work was partially supported by NIH Grant AI075219 and NSF Grants DBI-1060572 and IIS-1017181.

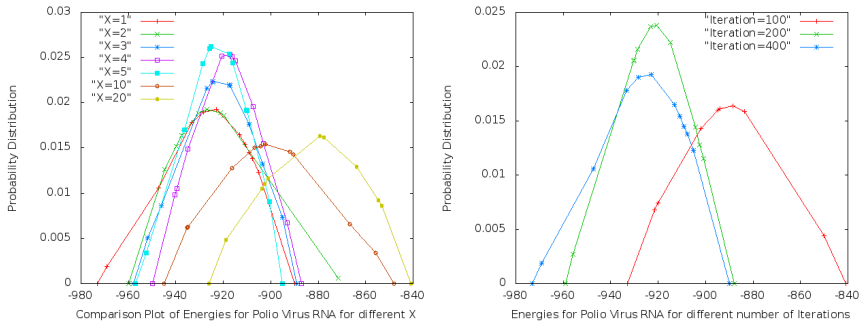

Figure 2.10: Comparison plot of the distribution of energy values of folded Polio virus RNA for different X (left); $X = 1, 2, 3, 4, 5, 10, 20$, and for different number of maximum iteration limit (right); $I = 100, 200, 400$. From the left figure, up to $X = 5$ (i.e. running Mfold every fifth iteration) performance of the algorithm is close to that for $X = 1$ (i.e. running Mfold every iteration), after that performance degrades. From the right one, energy optimization improves significantly when maximum iteration limit (I) goes from 100 to 200. When moving from $I = 200$ to 400, improvement is not that much.

Table 2.1: Summary of maximized and minimized sequence energies

Sequence	*Length*	*Initial dG*	$dG(Max)$	$dG(Min)$
Banana Virus1	405	-77.830	-142.43	-62.690
Pumpkin Virus	405	-91.570	-139.28	-83.780
Xenopus tropicalis hemoglobin	429	-111.19	-167.13	-98.520
Banana Virus2	531	-137.09	-189.56	-123.16
StCroix River virus	675	-191.50	-271.58	-160.57
GFP Jelly fish	720	-134.21	-243.56	-112.37
Citirus variegation Virus	849	-236.89	-333.42	-189.97
Honeysucklin	1035	-288.25	-434.46	-268.76
Potato Virus	1077	-299.97	-423.91	-249.43
Vibrio sp. Ex25	1260	-367.07	-500.60	-326.23
Polio Virus	2643	-666.68	-1003.2	-604.90

Chapter 3

Designing autocorrelated genes[1,2]

Transfer RNA is an adaptor molecule that acts as a bridge between the messenger RNA and the translated amino acid sequence. At one end, each tRNA carries an anticodon composed of three nucleotides that forms hydrogen bonds with a corresponding mRNA codon during protein synthesis. At the other end, tRNA is covalently bound to an amino acid. Each tRNA can be attached to only one amino acid, but as a consequence of the redundancy of the genetic code, several tRNAs may carry the same amino acid. Thus a one-to-many relationship exists between each amino acid and the tRNAs that code for it. This is related to the familiar triplet code, but more specialized since certain tRNAs match multiple different codons.

Because there are no physical differences between proteins translated using different synonymous codons, it is expected that synonymous codons would appear in roughly equal frequencies in the genome. However, in most genomes this is not the case. *Codon bias* is a long-observed phenomenon where the genes of a particular organism tend to favor the use of particular synonymous codons over others [44]. The degree to which the sequence of a given gene conforms to the preferences of the host is measured by the Codon Adaptation Index (CAI) [7]. Generally speaking, genes with higher CAI translate faster than genes with lower CAI [44]. For the purposes of synthetic biology, controlling protein expression has important implications in designing and fine-tuning gene regulatory networks. Controlling codon usage in a sequence is a demonstrated convenient and cheap method of accomplishing this, which can result in up to a 40-fold difference in expression [45]. Minimizing protein expression through

[1]R. Yeasmin, J. J. Tithi, J. Chen and S. Skiena. Designing autocorrelated genes. ACM-BCB, 2013

[2]Direct excerpts from [11]

altering codon usage is also an important application in gene synthesis, especially with regard to viral genomes [45, 46]. Beyond codon usage, factors such as codon-pair bias [9] and RNA stability also affect translation. However, the exact mechanism of optimal codon bias is still poorly understood, and there are many other considerations beyond the sequence level, such as rare codon and motif usage and mRNA structure. We refer the reader to the recent survey of Plotkin and Kudla [8] for the state of the current knowledge on coding sequence design to optimize protein expression.

Here, we will be concerned with a different factor that affects gene expression. Recent work shows that evolutionary pressure favors selecting repeating codons to code for successive occurrences of a given amino acid. Cannarozzi et al. [10] have demonstrated the effect of such autocorrelation in tRNA usage on gene translation speed. They mutated a green florescence protein (GFP) dimer to have respectively the most (and then least) autocorrelated coding sequence. The highly autocorrelated sequence translated and expressed up to 29% faster than the weakly autocorrelated one. The authors argue that autocorrelation in fact plays a more significant role than CAI index in identifying highly expressed genes. In [47], the authors proposed measuring the degree of tRNA re-usage in genes with an autocorrelation measure, tRNA pairing index (TPI). They proposed two variants of the index, TPI1 and TPI2 and showed statistically that genes with higher TPI values change their expression level rapidly. TPI1 is measured by computing the probability of the number of changes of tRNAs for each codon assuming constant codon frequencies (each codon is equally likely). TPI2, in contrast, is computed by assuming that the codon choice of amino acids is fixed (i.e., codon frequency for each amino acid is given). According to TPI1, a highly autocorrelated tRNA sequence is $TTTSSS$, which has the fewest number of tRNA transitions possible, and a highly anti-autocorrelated sequence that maximizes tRNA transitions is $TSTSTS$, where T and S correspond to different tRNAs expressing the same amino acid.

Although the tRNA Pairing Index (TPI) has been widely used to study the phenomenon of autocorrelated sequences [48], it counts successive transitions of tRNA usage without regard to how far they are on the sequence. Cannarozzi et al. [10] demonstrate that there is a distance aspect to autocorrelation, where the probability of finding identical adjacent synonymous codons decays with distance and both types of TPI measures fail to incorporate the distance effect in the scoring. Consider, for example, an amino acid sequence where *Leucine* residues occur at the 10th, 20th, and 400th position of the protein, to be coded by a mix of codons from two different tRNAs (say two of type A and one of type B). There are three relative orders in which these codons can appear:

AAB, BAA, and ABA. From the standpoint of TPI, both AAB and BAA are equally preferable, with only one transition, over the ordering ABA, which has two transitions. Yet from the vantage point of tRNA reuse, we would expect AAB to translate fastest, because the two occurrences of A are close-enough that the specific tRNA molecule employed in coding for the 10th residue is less likely to diffuse away before it can be used again. Indeed, through analysis on the genomes of several species, Cannarozzi et al. [10] have demonstrated that the degree of autocorrelation in codon ordering decays with distance.

Factoring in position distance is important, because the effects of autocorrelation or anti-autocorrelation at large distances become negligible as a previously-used tRNA molecule is more likely to diffuse away during a long stretch after it has been used. Furthermore, a tRNA transition may not prove costly if there quickly follows a second opportunity to reuse the previous tRNA. All in all, minimizing the number of tRNA transitions will generally not maximize the frequency of reusing tRNA molecules and will generally fail to optimize protein expression. This motivates us to consider a new autocorrelation measure (DICA) which weighs positional distance between codons as well the number of transitions.

In this research work, our major contributions are:

- *Distance-dependent measures of sequence autocorrelation* – We propose a new measure, Distance-Incorporated Codon Autocorrelation (DICA), which factors in the positional distance into evaluating the cost of a tRNA transition. Three types of reward models were considered based on threshold, inverse distance, and exponential functions respectively. The relative success of these various models on biological data may provide mathematical insight into the physical mechanism behind proper tRNA recruitment during protein translation.

  We believe that our DICA function better distinguishes the most significantly autocorrelated sequences than TPI does, and present the results of bioinformatics experiments to demonstrate this.

- *Designing optimally autocorrelated and anti-autocorrelated gene sequences* – The algorithmic problem of designing maximum (or minimum) autocorrelated coding sequence according to TPI for a given protein sequence and codon distribution is trivial: simply sort the available codons by tRNA compatibility, and then use them consecutively from the 5' to 3' ends of the gene.

  But gene design is much harder under our DICA criteria. We propose an exhaustive search procedure that exploits a new pruning criterion to design provably optimal designs for modest length genes, which when

coupled with heuristics performs very well on all sequences. An alternate search strategy and heuristics are proposed to design anti-autocorrelated sequences, and evaluated.

- *Mapping the autocorrelation design space* – How much freedom exists to rearrange the codons used in naturally-occurring gene sequences to modulate the degree of autocorrelation? We performed a study on all the genes in the yeast *Saccharomyces cerevisiae*, designing minimally and maximally autocorrelated genes using the wildtype (WT) codon distribution. We show that there is indeed considerable freedom to maximize the autocorrelation scores of a large number of genes in yeast, suggesting interesting experiments which synthesize our designs and compare them to wildtype.

3.1 Preliminaries

The phenomenon that different codons encode for the same tRNA and amino acid, yet are not seen in relatively equal frequencies, has been well established. However, the evolutionary advantages of codon bias are still being investigated widely [8]. Some advantages of using a certain synonymous codon frequency or distribution include proper mRNA folding and secondary structure, translation initiation and elongation rate, and translation accuracy and proper folding [8, 49, 50, 51]. Several different studies have already identified the scope of codon usage in its relation to translation speed. Tuller et al. [50] examined the contribution of low- and high- frequency codon usage to protein translation efficiency, and discovered that among most species, the first 30 to 50 codons of a given sequence tend to favor rarely used codons, before a "ramp" into a period where commonly-used codons dominate. Interestingly, both the available tRNA pool and the genome sequences co-evolve to maintain this profile, suggesting that genome sequences evolve to be deliberately slowly translated directly after translation initiation.

Cannarozzi et al. [10] further corroborated the importance of rare and frequent codon positioning in translation speed. Synonymous codon usage is not random and the codons are not equivalent [10, 52, 53]. In particular, the frequency of codons in gene sequences correlates well to the frequency of the tRNA molecules corresponding to these codons [52, 54]. If a recently used tRNA diffusion were slower than the ribosomal progression, then it would be efficient to re-use the same tRNA for subsequent occurrences of the same amino acid. Furthermore, genes, especially those that must be translated rapidly under stressful conditions, appear to have evolved to increase autocorrelation

of codons, having higher autocorrelation measure than similar sequences that have randomly shuffled all of their synonymous codons.

A pattern of codon usage may either be induced due to mutational processes or by synonymous mutations which natural selection may favor. Mutational processes, such as a silent mutation changing one synonymous codon to another has no overall impact on the fitness of a species. However if natural selection plays a role in how that mutation affects fitness, then a similar codon usage pattern could be seen across entire genomes or species [8]. Positive correlation between codon bias and a gene's expression level has been observed among different species [6, 7, 8, 55]. Conversely, the synonymous codon substitution rate among diverging species is negatively correlated with gene expression level [8]. An extreme codon bias has been observed in highly expressed genes to match a skew in iso-accepting tRNAs (tRNAs that carry the same amino acid). Xu et al. [56] demonstrated that highly expressed genes have more synonymous codon usage biases originated from selective pressure, however this is probably a species-specific phenomenon and other organisms evolve codon bias more for translation efficiency. Qian et al. [57] hypothesizes that synonymous codons are translated with similar speeds under the codon-tRNA balance optimized by nature to improve translational efficiency. Ikemura [52] further shows an existing strong positive correlation between codon usage and tRNA content in both *Escherichia coli* and *Saccharomyces cerevisiae*, which is dependent on an individual gene's protein production levels.

In [47], the authors proposed measuring the degree of tRNA re-usage in genes with an autocorrelation measure, tRNA pairing index (TPI). They proposed two variants of the index, TPI1 and TPI2 and showed statistically that genes with higher TPI values change their expression level rapidly. TPI1 is measured by computing the probability of the number of changes of tRNAs for each codon assuming constant codon frequencies (each codon is equally likely). TPI2, in contrast, is computed by assuming that the codon choice of amino acids is fixed (i.e., codon frequency for each amino acid is given). According to TPI1, a highly autocorrelated tRNA sequence is $TTTSSS$, which has the fewest number of tRNA transitions possible, and a highly anti-autocorrelated sequence that maximizes tRNA transitions is $TSTSTS$, where T and S correspond to different tRNAs expressing the same amino acid. Cannarozzi et al. [10] demonstrate that there is a distance aspect to autocorrelation, where the probability of finding identical adjacent synonymous codons decays with distance and both types of TPI measures fail to incorporate the distance effect in the scoring. Codon bias is already an important consideration when designing genes, and matching synonymous codon frequencies of a gene sequence to the frequency of the host genome generally provides optimal translation results

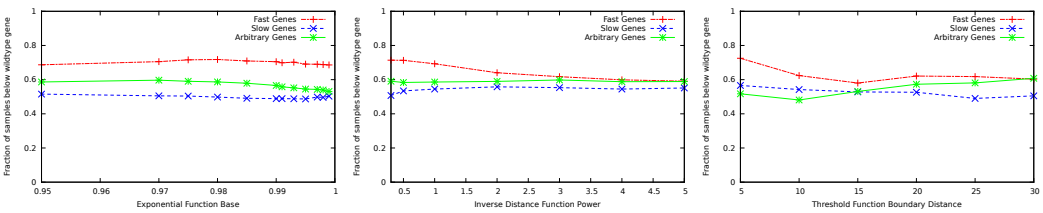

Figure 3.1: Comparison of different optimization functions to identify the best differentiation between fast vs. slow genes. We checked 90 wildtype genes of different speeds, where the tRNA abundance in the cell for each gene is used as a measure of speed, i.e., larger the number of transcripts, more the gene is expressed. 1000 random samples were generated for each wildtype gene, by random swaps of tRNAs of different types corresponding to the same amino acid. *Left:* the fraction of random samples with score less than wildtype sequence for different values (0.950 to 0.999) of exponential function base. *Middle:* different exponents (0.2 to 5) for the inverse distance function. *Right:* different threshold values (5 to 30) for the threshold function. In the range of 0.98 to 0.99, the exponential function score shows maximum separation in different types of genes. For the inverse distance function, an exponent of 0.3 shows the best result. For the threshold function, a threshold value of 20 to 25 shows good results. Among the three types of functions, the exponential function shows the best result, as it can successfully differentiate the fast, slow and average speed genes for all base values in the experiment.

for all but the highest-expressing genes [45]. We consider a new autocorrelation measure - DICA, which accounts for tRNA transitions as well as distance between the synonymous tRNAs.

3.2 Distance-incorporated codon autocorrelation

The 'Distance-Incorporated Codon Autocorrelation' (*DICA*) is our proposed metric of gene autocorrelation. DICA is calculated for a coding sequence by finding positions of all synonymous codons for a given amino acid and summing a function, $F(d(i,j))$ (the reward function), which assigns a positive score based on the distance between the synonymous codons. Here $d(i,j)$ is the distance between codons translated by the same tRNA i.e., if a tRNA repeat is found at position i and j, then the distance between these two is $d(i,j) = j - i$. Since there are nine amino acids that have synonymous codons translated by different tRNAs (*A, R, G, I, L, P, S, T, V*), this process is repeated for each of those nine amino acids. The final DICA is obtained by

summing up the contributions from each amino acid and then normalizing the sum by the sum of the scores of maximum possible changes of the tRNAs corresponding to each amino acid.

More formally, suppose amino acid a occurs n_a times in the sequence S at positions $P_1^{(a)}, \ldots, P_{n_a}^{(a)}$, with the tRNAs in these positions being $T_1^{(a)}, \ldots, T_{n_a}^{(a)}$. The DICA ($D_S$) for S is defined as

$$D_S = \frac{\sum_{a=1}^{K} \sum_{i=1}^{n_a} \sum_{j=(i+1)}^{n_a} \theta(T_i^{(a)}, T_j^{(a)}) \times F(d(P_i^{(a)}, P_j^{(a)}))}{\sum_{a=1}^{K} \sum_{i=1}^{n_a} \sum_{j=(i+1)}^{n_a} F(d(P_i^{(a)}, P_j^{(a)}))},$$

where $\theta(T_i^{(a)}, T_j^{(a)}) = 1$ if $T_i^{(a)} = T_j^{(a)}$ and 0 otherwise. The value of D_S ranges from 0 to 1 and it increases with the increase of autocorrelation in the sequence.

Because autocorrelation appears to decay slowly with distance [10], for a given synonymous codon, the probability that the next codon is the same decreases as the distance increases. To determine the extent to which autocorrelation is visible and how it decreases over distance, we considered three different distance functions to calculate the DICA of genes:

- *Threshold function* – One possible candidate distance function is threshold function, where the autocorrelation effect is visible equally at all positions up to a certain distance, d_t, after which it is not visible at all. Here up to a certain maximum distance each tRNA repeat is rewarded by a fixed amount, after which no reward is given. Our reward function is 1 for a repeat within the boundary, i.e., $F(d(i,j)) = 1 \; if \; d(i,j) \leq d_t$ and $F(d(i,j)) = 0 \; if \; d(i,j) > d_t$.

- *Inverse distance function* – In an inverse distance function, each tRNA repeat is rewarded by the inverse of the distance between the positions of those two tRNAs in the original amino acid sequence, i.e., if the positions of the two tRNAs are i and j, then the reward for the repeat is $F(d(i,j)) = \frac{1}{(j-i)^p}$, where p is the power of the function which is a parameter that we need to optimize. Here as the distance between two codons increases, the effect of autocorrelation diminishes with distance. Different powers to the inverse of the distance value were examined to get the best distance measure.

- *Exponential function* – An exponential distance function is another measure to take the impact of distance into account. Here the effect of distance on autocorrelation decreases exponentially with the increased distance between tRNA repeat. We used the function of the form

$F(d(i,j)) = c^d$, where c is a constant base value and d is the distance of the tRNA repeat. An optimal value of c was also examined.

Any of the above scoring functions could in principle be used to calculate an autocorrelation score. We evaluated each of these three distance functions, as well as TPI_S, our TPI estimate, to investigate which model best accounts for the observed selective pressure of codon autocorrelation to time-sensitive expressed genes. In general, TPI is defined as 1 minus twice the percentile of the number of tRNA transitions over all possible combinations of a sequence. If the percentile of the number of tRNA transitions is C, then the corresponding TPI will be $1-2C$ [47]. So a TPI of 1 indicates that a sequence is at the highest percentile of minimizing codon transitions, i.e., no other permutation of the sequence will have fewer transitions than it. Conversely, a TPI of -1 indicates the sequence is at the lowest percentile of minimizing codon transitions, where no sequence will have more transitions than it. A score of 0 has the median number of transitions. For example the sequence AAABBB has TPI close to 1, as there is one transition, and only two out of $\binom{6}{2}$ sequences will have one transition. Likewise the sequence ABABAB has TPI close to -1, since it has 5 transitions, and only one other sequence (i.e., BABABA) will have that many transitions. Every other sequence has fewer transitions.

In our experiments we approximate the TPI score of a given sequence by counting all synonymous tRNA transitions for each amino acid having multiple synonymous tRNAs and then normalizing the sum by the sum of the frequency of each of such amino acids. We then subtract this value from 1 to get the final approximate score. The formal definition of our approximate version of TPI is as follows:

Consider a sequence S of length L composed of K different amino acids with more than one synonymous tRNA. Each amino acid a occurs n_a times in the sequence at positions $P_1^{(a)}, \ldots, P_{n_a}^{(a)}$. Let, the tRNAs in these positions are $T_1^{(a)}, \ldots, T_{n_a}^{(a)}$. Now our simulated TPI (TPI_S) score can be explained as

$$TPI_S = 1 - \frac{\sum_{a=1}^{K} \sum_{i=1}^{n_a} \tau(T_i^{(a)}, T_{(i+1)}^{(a)})}{\sum_{a=1}^{K} L^a}$$

where $T_i^{(a)}$ is the tRNA at position i corresponding to amino acid a and $T_{(i+1)}^{(a)}$ is the tRNA at the next position of the same amino acid, $\tau(T_i^{(a)}, T_{(i+1)}^{(a)}) = 1$ if $T_i^{(a)} \neq T_{(i+1)}^{(a)}$ and 0 otherwise.

We use TPI_S since calculating TPI is inherently recursive and thus prohibitively time consuming for even moderately long sequences. Since our analysis uses the ranks of each measure rather the absolute score, TPI_S should be a

sufficient estimate for verifying how accurate counting tRNA transitions is for predicting expression. Since TPI_S only considers codon transitions without accounting for their frequencies, it better corresponds to TPI1. We estimated TPI1 rather than TPI2 because both TPI1 and DICA measures are independent of the genome codon frequency and do not take any other biases into account. Later in the paper we discuss accounting for background codon frequency as an inherent optimization problem parameter.

We studied two data sets of yeast genes, with around 4480 genes in the first set and 5018 genes in the second set. We performed an experiment on the 200 fastest and 200 slowest genes from both of these sets. The first set (referred to as *set 1* hereafter), was collected from the WWW site [cel] under the α-factor/cell cycle arrest microarray experiment ([58], [59]). The second data set (*set 2*) was ranked based on transcript abundance of mRNAs available in the cell for all yeast genes [60]. Genes that express abundant mRNAs are presumably being transcribed under time pressure, so a gene that needs to be translated quickly would likely have high mRNA counts. It is readily accepted that the number of RNA transcripts is a measure of gene expression (indeed, this is what drives microarray technology and RNA-seq) and generally correlates with protein abundance. The other part of the argument is that highly expressed genes should evolve to translate quickly. This is the presumption behind the codon adaptation index (CAI), [7] which is generally used as a proxy for gene expression in yeast.

3.2.1 Parameter optimization

We have optimized the parameters for these candidate DICA scoring functions, to identify which would best predict the translation speed of naturally observed sequences. Our experimental results (fig. 3.1) suggest that the exponential function best predicts the codon usage pattern to optimize translation speed.

For the exponential function, we investigated the optimal value of the proper exponential base c. In general, values from 0.98 toward 0.99 tend to show the best results. The inverse distance function also follows the observation that autocorrelation decays with distance to some extent as the reward for tRNA repeat decreases with the increase of distance between two tRNAs. We found empirically that an exponent of 0.3 for the polynomial function best explains the autocorrelation.

For the threshold function, a threshold between 15 and 30 is best. We recommend a threshold of 20. For a threshold above 15, the DICA of the tRNAs is visible. The threshold should not be larger than 30 because the function will not properly differentiate genes of different speeds. Note that the threshold function scores all tRNAs similarly up to a certain distance.

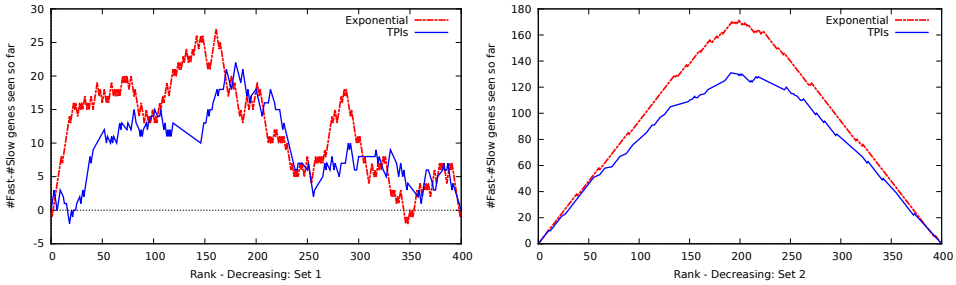

Figure 3.2: Comparing the exponential vs. TPI_S scores for 200 fast and 200 slow genes selected from all the available genes for two different data sets. We merged these fast and slow genes and sorted by scores from highest to lowest, shown along the X-axis. The Y-axis shows the difference between the number of fast genes and slow genes seen so far (with scores $\geq$ current gene score). The red lines are for the exponential DICA scoring function and the blue lines are for TPI_S scores. If fast genes are more autocorrelated than slow genes, then scores for fast genes should be higher. Hence, during the first half of the interval, the trend of the curve should go up and then go down at the later half. Genes from both of the data sets we used completely agree with this expected behavior for both DICA and TPI_S scores. However, DICA explains the behavior better than TPI_S since it shows a higher upward trend at the beginning, indicating it better predicts scores of fast genes.

After that, no reward is given for the repeat. This is more stringent than what actually occurs in nature. Fig. 3.1 shows the veracity of the parameter optimization process.

3.3 Evaluation

Since our experimental results demonstrate that the exponential scoring function for DICA best correlates with translation speed (fig. 3.1), we calculated exponential DICA and TPI_S scores for 400 genes (200 fast and 200 slow) from both of the data sets explained earlier and then for each set sorted the genes from maximum to minimum score. Next we iterated through the 400-item rank-ordered list of genes in both measures, plotted the difference of the number of fast genes and the number of slow genes at each rank position. Fig. 3.2 demonstrates that in general, DICA is more successful than TPI_S in differentiating fast and slow genes for both of these data sets. We performed *Mann-Whitney U* for each of the data sets. The p-values obtained for

the DICA and TPI_S for *set 1* were respectively 0.025 and 0.136 and corresponding p-values for *set 2* were $2.03e^{-57}$ and $2.53e^{-39}$. For *set 1* DICA could differentiate fast and slow genes significantly ($p < 0.05$) whereas TPI_S could not. For *set 2* though both scores performed good, DICA was more accurate in differentiating fast vs slow genes.

3.4 Designing DICA optimized genes

Given an amino acid sequence with frequencies of different tRNAs that can be used, we seek to design the gene with the maximum (or minimum) DICA that follows the given tRNA distribution. Previous studies have shown that simply maximizing CAI or always using the most frequently occurring tRNA does not maximize expression [45]. As an important application, the tRNAs used in the algorithm for de novo genes could therefore correspond to the host background tRNA levels, as this balances codons and their tRNA molecules presumably to optimize translational efficiency [46], or correspond to the original gene frequency in an already existing gene, maintaining the sequence's original codon bias. We propose both optimal and heuristic strategies to design highly autocorrelated and anti-autocorrelated genes. We use depth-first search (DFS) with pruning to get the optimal solution. Our optimal algorithms can successfully predict the most and least autocorrelated genes for modest length sequences. We also propose heuristic algorithms which, when coupled with the optimal ones, perform very well on sequences of any length and run very fast.

3.4.1 Highly autocorrelated gene design

An algorithm for optimizing TPI greedily minimizes the number of tRNA transitions, but this might not lead to a globally optimal DICA solution when distance is included. Our algorithm finds the optimal solution by minimizing the distance between synonymous tRNAs corresponding to same amino acid globally. Algorithm 3 gives a sketch of our backtracking based DICA optimized gene design strategy to find the most autocorrelated sequence.

In order to prune unnecessary paths during the search, we pre-calculate the score for potential solutions at every step. For eliminating sub-optimal branches, two different pruning strategies have been used: *strategy 1* is obvious but gives a weak bound, and *strategy 2* gives a better bound for pruning. The strategies are described below:

- *Strategy* 1*:* – suppose the given sequence is of length L. At the current step we have already calculated up to length $L' < L$. Now, we check

Algorithm 3 OptimalSearchMax (Input: Amino Acid sequence of length L, tRNA frequencies; Output: Most autocorrelated sequence)

```
insert the root node with empty sequence into STACK
while STACK is not empty do
  remove a node from the STACK
  if the sequence is of length L then
    calculate DICA for the sequence
    if current DICA is better than max DICA then
      make the current sequence most autocorrelated sequence and update
      max DICA
    end if
  else
    check the current sequence for pruning
    if current sequence cannot be pruned then
      insert the sequence with all possible tRNAs at the next position into
      STACK
    end if
  end if
end while
```

Algorithm 4 HeuristicMax (Input: Amino Acid sequence of length L, tRNA frequencies; Output: Most autocorrelated sequence)

```
while there are more amino acids, for each do
  while there is more available tRNA do
    find the tRNA (T) with maximum frequency (f) from currently available
    set of tRNAs
    calculate distance between each consecutive tRNA position for current
    amino acid
    find the run of f consecutive positions (window W of size f) with mini-
    mum distance
    place all the tRNAs of type T at current window
    remove the positions in window W from available set of positions, re-
    move T from available tRNA list
  end while
end while
```

all the available tRNAs to place the one at position $L'+1$ that gives the best DICA with the sequence from 1 to L'. Then, we assume the same tRNA at all the remaining positions, and the score for the current sequence is the score of the sequence $1..L'$ *plus* the score for sequence $L'+1..L$, where all tRNAs are assumed to be of same kind, and the interacting score of these two parts, i.e., for each tRNA from segment $L'+1$ to L, the contribution of each tRNA from segment 1 to L'. If the calculated score is greater than the best observed sequence (with maximum autocorrelation) encountered so far, then the new sequence is a potential candidate to explore, and the current sequence with all possible tRNAs for the next position are inserted into the DFS stack list. Otherwise, it is pruned.

This strategy is correct, as we never underestimate the total reward for the potential sequence to be encountered later while pruning. At position $L'+1$, we place the tRNA with maximum DICA with the already encountered part among all the available tRNAs, and we are assuming the same tRNA at all subsequent places. Hence,the score can be no better than our calculated score with $1..L'$ at the beginning, irrespective of the types of tRNAs at positions $L'+1..L$.

- *Strategy* 2 – here we try to improve the upper bound. Again, for a sequence of length L, we have optimal tRNA placement up to position L'. We check the score at position $L'+1$ for all available tRNAs with the sequence $1..L'$, and place the one with the best score and decrease the available tRNA frequency of that type by one. Similarly, we check the position $L'+2$ and continue this way until we reach the end of the sequence. The sum of all these rewards gives the interaction score of the yet to calculate part with the already calculated part. To calculate the score for the unknown part of the sequence, we assume all tRNAs are of the same kind for a maximally autocorrelated gene. Finally, we add the two scores with the score of the already calculated part, which gives the final sequence score. Then, we compare this score with the score of the most autocorrelated sequence observed by that time. If the current score is better, we explore the branch, otherwise it is pruned.

The correctness of *strategy* 2 follows from *Lemma 1.*

Lemma 1. *Let $f(d)$ be any monotonically non-increasing function with values in the range $[0,1]$. Let $S(t,d)$ be a scoring function (with t being several possible options available at hand and d being a state or position) that satisfies $S(t,d') = f(d'-d) \times S(t,d)$. If $S(t_1,1) \geq S(t_2,1)$, then $S(t_1,1)+S(t_2,2) \geq S(t_2,1)+S(t_1,2)$.*

Proof.

$$S(t, d') = f(d' - d) \times S(t, d)$$
$$\Rightarrow S(t_1, 2) = f(2-1) \times S(t_1, 1)$$

and

$$S(t_2, 2) = f(2-1) \times S(t_2, 1)$$

If $f(2-1) = 0$ then we have $S(t_1, 1) \geq S(t_2, 1)$ which is true by definition.

Again, if $f(2-1) = 1$ then $S(t_1, 1) = S(t_1, 2)$ and $S(t_2, 1) = S(t_2, 2)$. Then the equality condition is satisfied.

Now if $0 < f(2-1) < 1$ then,

$$S(t_1, 1) \geq S(t_2, 1)$$
$$\Rightarrow S(t_1, 1) - f(2-1) \times S(t_1, 1) \geq S(t_2, 1) - f(2-1) \times S(t_2, 1)$$
$$\Rightarrow S(t_1, 1) - S(t_1, 2) \geq S(t_2, 1) - S(t_2, 2)$$
$$\Rightarrow S(t_1, 1) + S(t_2, 2) \geq S(t_2, 1) + S(t_1, 2)$$

Hence, for any value of $f(2-1)$ in the range [0-1], $S(t_1, 1) + S(t_2, 2) \geq S(t_2, 1) + S(t_1, 2)$ is true. □

Suppose, we are trying to prune the search at position $L' + 1$. Now, at $L' + 1$ we greedily place the tRNA with maximum score, then at $L' + 2$ and so on till L. Based on *Lemma 1* we can say our greedy decision at every node of the tree will always let us find the sequence from $L' + 1..L$ of maximum score, given the already obtained sequence $1..L'$, as we are never violating the monotonously decreasing order of the reward function.

Our optimal algorithm can correctly find the most autocorrelated sequences by searching for the optimal solution of each possible amino acid independent of others, and then, merging the solutions together to get the final optimal sequence. However, the complexity of the algorithm increases exponentially with the increase of the number of an particular amino acid in the sequence. Hence, we apply the heuristic algorithm to parts of the sequences, where an amino acid frequency is very high. Algorithm 4 describes our heuristic approach for finding the suboptimal solution. Our heuristic algorithm runs in quadratic time and the generated sequence is almost as good as the optimal one.

3.4.2 Anti-autocorrelated gene design

We are also interested in designing anti-autocorrelated genes. To find the anti-autocorrelated genes the TPI approach would try to maximize the num-

ber of synonymous tRNA transitions. We designed an algorithm that gives the least autocorrelated sequence by maximizing spacing between synonymous tRNAs throughout the sequence. Here, our approach is similar to the one for maximizing autocorrelation, DFS with pruning. However, instead of maximizing the autocorrelation we minimize it by placing synonymous tRNAs further apart. Likewise, we developed an heuristic algorithm to get suboptimal but fast solutions for minimally autocorrelated sequences. The heuristic approach for designing genes with least autocorrelation is described in algorithm 5.

Algorithm 5 HeuristicMin (Input: Amino Acid sequence of length L, tRNA frequencies; Output: Least autocorrelated sequence)

```
while there are more amino acids, for each do
    sort current available positions of tRNAs based on distance between consecutive positions
    if there is only one type of tRNA (T) then
        return current optimal tRNA sequence with all Ts.
    else
        place different tRNAs alternately one after another starting from the smallest consecutive distance
        decrease tRNA frequencies from the available tRNA list based on the tRNAs placed at previous step
    end if
end while
```

Both of these heuristic algorithms run in quadratic time on the number of available tRNAs for each amino acid.

3.4.3 Merged search/ heuristics algorithms for gene design

We integrated our search based algorithmic approach with our heuristic approach to give a combined algorithm. This algorithm follows the optimal approach in cases where the solution can be found within a good enough runtime, otherwise it will use the heuristic approach to get a fast solution.

Based on our experimental results, we decided when to use which strategy. In particular, when an amino acid is repeated more than 20 times and more that 2 tRNAs are available for it, the algorithm takes quite long to find the optimal solution. For example, for a sequence having the amino acid 'L' 20 times where 3 different tRNAs were used, it took a few minutes to get the optimal solution. However, for a sequence where the amino acid 'L' was repeated 28

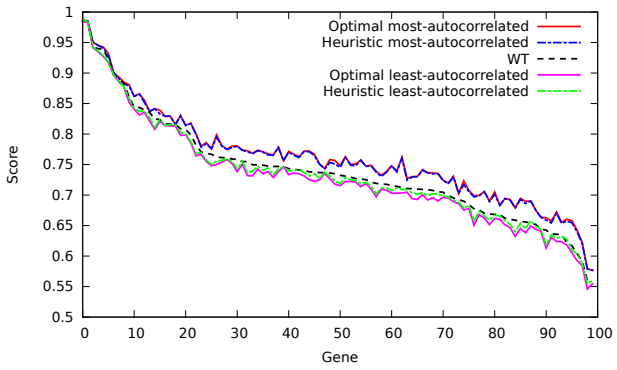

Figure 3.3: Comparing Optimal and Heuristic function scores with wildtype (WT) scores for most and least autocorrelation. The red and magenta lines indicate the Optimal most and least autocorrelation scoring function, while the blue and green lines indicate the Heuristic most and least autocorrelation scoring functions and the black line (WT) is for wildtype scores. The X-axis shows different gene positions and the Y-axis shows corresponding DICA values.

times and 4 different tRNAs were available for it, the algorithm took around an hour to find the desired sequence. We checked for several other cases and decided to apply the heuristic strategy on amino acids in the sequence where it is repeated more than 20 times and more than two tRNAs are available for it. Otherwise we apply the search based approach.

Our algorithms always give the optimal solutions. However, to minimize the run time, we proposed heuristic algorithms that give good solutions for sequences of any length with much faster runtime, quadratic in the length of sequence. Fig. 3.3 shows that the performance of our heuristic algorithm is comparable to the optimal solution for both most and least autocorrelated sequence design. Here, we selected 100 wildtype genes and sorted these genes from maximum to minimum by their DICA values. The black line shows the trend of wildtype gene scores. Then we ran both of our optimal and heuristic algorithms on these genes. The trend for the optimal most (least) autocorrelated sequence scores of these genes are shown in red (magenta) line, and the maximized and minimized results according to the heuristic algorithms are shown in blue and green.

3.4.4 Algorithm validation through experiments

We applied our algorithms on several genes to verify experimentally that our algorithms are performing well. We selected several yeast genes and generated

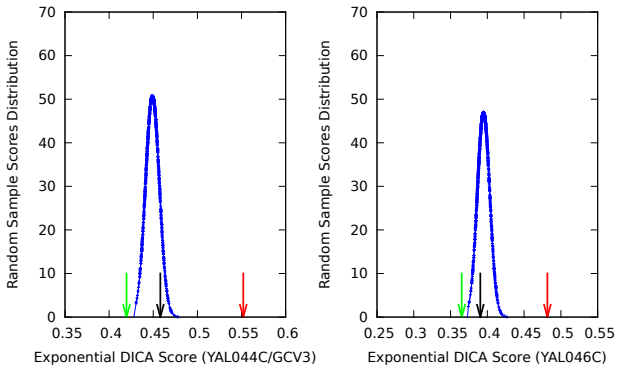

Figure 3.4: Distribution of DICA values of 1000 randomly generated samples for two yeast genes *YAL044C* (or *"GCV3"*) and *YAL046C* are shown in blue, the designed DICA optimized maximum sequence score is indicated by red arrow, the green arrow indicates the optimized minimum sequence score, and the black arrow is for the wildtype gene's DICA. Here the score of the designed maximum sequence is greater than all the randomly generated sequences for both of the genes, while the minimum score is less than all of the randomly generated sequence scores.

1000 random samples for these genes by random swap of tRNAs corresponding to the same amino acid, maintaining similar tRNA distribution. Then we generated DICA optimized (or de-optimized) gene sequences using our search based algorithms. Later we compared the scores of these random sequences with the score of the designed sequence. We observed that, our designed sequence score is always greater (or less for anti-autocorrelated sequences) than the random sample scores and the wildtype score. This is an indication that our algorithms generate sequences with optimal scores. Fig. 3.4 shows the distribution plot of the DICA scores for 1000 random samples along with the wildtype and the generated most and least optimal sequence scores for two genes, *YAL044C/GCV3* and *YAL046C*. It is clearly visible in the figures that our DICA optimized sequence scores are further apart from the random sequence score distribution, while the wildtype score is within the distribution.

Our optimal algorithms can perfectly predict the most and least autocorrelated genes. Fig. 3.4 shows results for two such genes. For both of these genes we generated 1000 random samples, and compared the scores of these random samples with our DICA optimized scores. The distribution of the scores of these random samples were lower than our designed most autocorrelated gene score and higher than the least autocorrelated one.

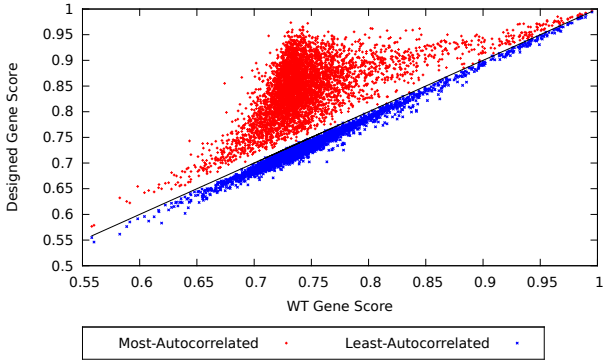

Figure 3.5: The figure compares combined Optimal and Heuristics function scores for 5018 yeast genes with wildtype (WT) scores for most and least autocorrelation function. Red dots are for wildtype vs (Optimal+Heuristic) most autocorrelation score, blue dots are for least autocorrelation score, black line indicates wildtype vs wildtype scores. Here, higher the height along Y-axis, more autocorrelated is the sequence. The figure shows, red dots are always above the black line and blue dots are below the line, having marginal difference in cases where either wildtype gene itself is significantly auto/ anti-autocorrelated or there is little room to design.

3.5 Results and discussion

3.5.1 Data collection and processing

As mentioned before, we used two different data sets for our experiments. For the first set we took gene expression data from the WWW cite [cel] under the α-factor/cell cycle arrest microarray experiment (Cho et.al. [58] and Spellman et al. [59]). The data had gene expression levels for over 4000 genes from the beginning of the experiment to 119 minutes, taken every 7 minutes after arrest.

The speed of translation of each mRNA was estimated the same way as Cannarozzi et al. [10] did by taking the difference of the normalized log2 of each expression level between every two time intervals and averaging the positive differences in expression levels. Genes with incomplete expression data were excluded. We use the following equation for calculating the speed of a gene from the microarray data:

$$Speed = \frac{\sum \left(E_{(t+1)} - E_t\right)}{n}$$

where E_t is the value of expression at time stamp t, $(E_{(t+1)} - E_t) > 0$ and $n =$

number of changes for which $(E_{(t+1)} - E_t) > 0$. However, rather than taking all the positive differences, we took the average of the top three differences to determine the speed which also adequately express the speed of genes.

The second set (set 2) of yeast genes were taken from [60] where we used the transcript abundance of mRNA in the cell as a measure of speed. Translation speed has been historically very difficult to explicitly measure, and in the absence of good metrics, mRNA transcript count has been used in one of our experiments as a good proxy for translation speed. Specifically, mRNA count can account for up to 40% of translation product [61]. In yeast, mRNA transcript count and protein output correlate with an r^2 value of 0.59 [62]. Hence, mRNA count should account for a valid approximate measure of translation speed. Genes having the largest number of transcripts (in the range 538.52 to 12336.15) available in the cell were taken as fast genes. Genes with the number of transcripts ranging from 0.02 to 1.13 were taken as slow genes. The RNA sequences corresponding to genes of these data sets were collected from Saccharomyces Genome Database (www.yeastgenome.org).

3.5.2 Freedom of design

To evaluate the design landscape imposed by our DICA scoring function, we determined the most and least autocorrelated genes for all the yeast genes. In our final algorithm we merged the search based and heuristic approaches together. We applied our merged algorithm on 5018 yeast genes to find the most and least autocorrelated sequences. Fig. 3.5 shows the performance of our algorithm by comparing the DICA of our algorithm generated sequences with wildtype sequence score. Our algorithms optimize all of these genes to give the most or least autocorrelated sequences based on their synonymous codon distributions. For some of these genes, the optimized scores are largely different compared to their corresponding wildtype scores. We analyzed the top 100 of these genes. These are mostly very long genes with large variation in their tRNA use. Note that, some of the wildtype genes, of which most are highly expressed genes, are also highly autocorrelated. These genes evolved to come close to the optimal solutions. There was little room to further optimize these genes. Our algorithms could successfully reach that optimization level.

3.6 Conclusion

In this paper, we have proposed a new distance dependent autocorrelation measure, DICA and proved that autocorrelation effect on genes is distance dependent by using our newly developed scoring method. So far previous

studies worked on autocorrelation, where only the number of tRNA changes were considered to measure autocorrelation and were minimized (maximized) to design highly autocorrelated (anti-autocorrelated) genes. We used our new scoring method to further optimize (or de-optimize) the sequence which might result in higher (lower) levels of gene expression.

We also show that, in the absence of a tRNA charging and translational model autocorrelation is well explained by exponential decay, though we emphasize that we do not intend that the actual mechanics of translation is explained by this function. We have proposed algorithms for designing most and least autocorrelated genes. However with the increase of the length of the amino acid sequence the algorithm complexity grows exponentially and it becomes a problem when an amino acid appears more than 20 times in the sequence and more than 2 synonymous tRNAs are available for that amino acid. To overcome the problem we propose heuristic algorithms which sometimes may give suboptimal solutions but run in quadratic time. Our combined (optimal and heuristic) algorithms run very fast and give solutions comparable to the one where we apply only the optimal approach. We ran our combined algorithm for 5018 yeast genes to find the most and least autocorrelated sequences of these genes.

It is important to note that while we demonstrated DICA's significance in predicting gene translation speed, and showed evidence that selective pressures will want to group synonymous codons in time-sensitive genes closer together, for the purposes of synthetic biology DICA is far from the only concern in designing an optimal gene. DICA optimized genes may not necessarily result in the highest level protein expression, but may act as an important parameter along with other optimization criteria to maximize (or minimize) protein expression. One important future direction would be to take the highly expressed wildtype genes and recreate the sequence optimizing DICA to see its effects on expression rates. Other considerations are also important for proper gene design, and the algorithm discussed in our paper may be combined with other considerations such as avoiding certain nucleotide sequences, a consideration that Welch et al. [45] uses in their algorithm. Finally, comparing DICA and TPI_S scores by excluding the first 50 codons would lend greater credibility to the importance of optimizing codon autocorrelation in evolution, as these codons are translated slowly [50].

3.7 Acknowledgement

We note that the second and third authors contributed equally to the design and setup of this work. We specially thank Bruce Futcher for helping us

with yeast gene mRNA transcript abundance data collection. This work was partially supported by NIH Grant AI075219 and NSF Grants DBI-1060572 and IIS-1017181.

Chapter 4

Statistical analysis of tiled microarray gene expression data[1]

Microarray data analysis is an continually evolving technology for gene expression analysis and for identifying biological processes, functions and activities affected by genes and discovering novel diseases. One particular challenge with this procedure is the amount of data it generates. Analyzing microarray data generally consumes considerably more time than the laboratory protocols required to generate the data [63]. Part of the challenge lies in assessing the quality of the data and normalizing the data in one array in a way to make it comparable to data from other microarray experiments.

We use microarray based genome expression profiling to create a global picture of the cellular function and to compare genomic expression changes at different conditions. Through the analysis of the microarray data, we aim to identify novel housekeeping genes and differentially expressed genes. Based on the expression changes across the array, we seek to cluster genes at different experimental conditions. We further work on gene ontology enrichment analysis for a set of genes of our interest.

Viruses use various mechanisms to avoid antiviral responses introduced by the host cell. One commonly used mechanism is to inhibit host gene expression, while selectively sparing viral genes. *Gammaherpesvirus* infection at the lytic stages of the host cell causes widespread degradation of cytoplasmic mRNAs through the activity of the viral endonuclease SOX [64]. However, the impact of the SOX-induced infection on viral mRNAs had not been explored yet. Our analyzed data on *Murine gammaherpesvirus 68* (MHV68) was used to study

[1]This chapter contains materials from [4].

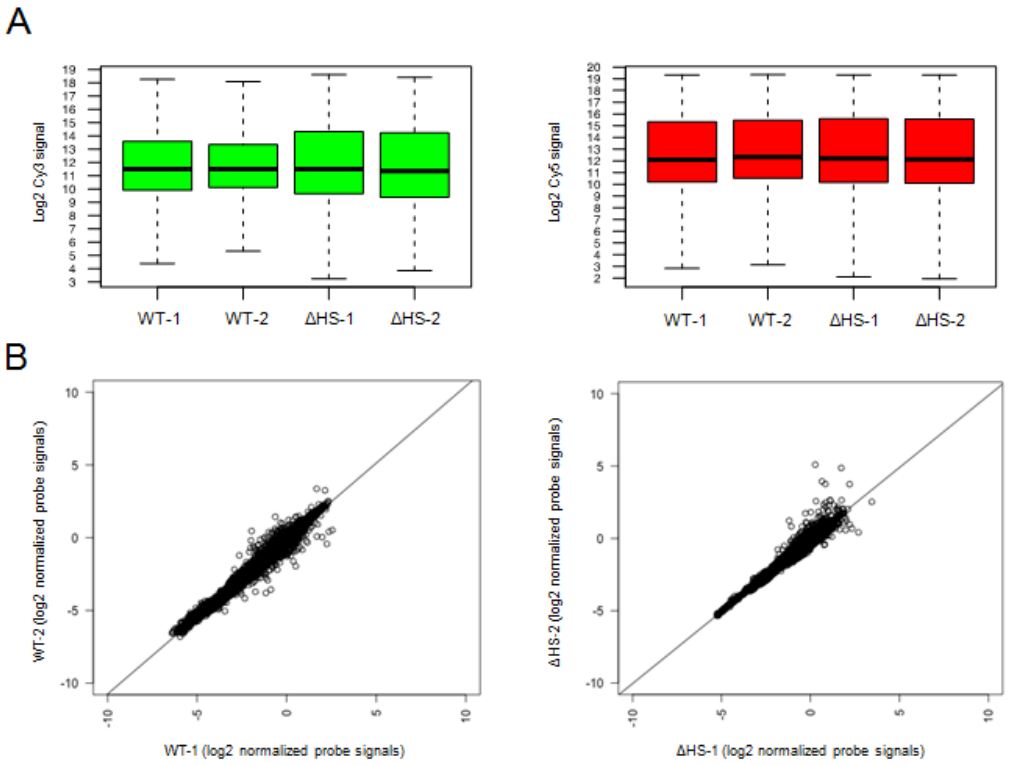

Figure 4.1: Quality control experiments on array 5 data. Figure-A. Box-and-whisker plot (ignoring the outliers) of cy3 (green) and cy5 (red) labeled two independent biological replicates at different experimental conditions, infection with WT or ΔHS viruses. Figure-B. Scatter plot of wildtype vs wildtype and mutant vs mutant.

the changes in viral gene expression in the presence or absence of the viral Sox during infection [4]. Interesting results came out of the analysis. The virus employs a strategy to degrade host cell mRNA level, which affects viral mRNAs as well. Further study revealed that, the viral mRNA degradation is an important step to control the overall viral gene expression. For the analysis, we used microarray gene expression data from two different mutants of MHV68 infection - one that is impaired for host shutoff (ΔHS, host shutoff defective) and the other one is the rescue virus (MR/WT) in which the mutation was restored to wildtype. Analysis outcome revealed that, MHV68 SOX protein down regulates the majority of viral transcripts at all kinetic classes [4].

The type of the data-sets used in the microarray analysis and the quality check experiments are described in sec. 4.1. Section 4.2 includes our results

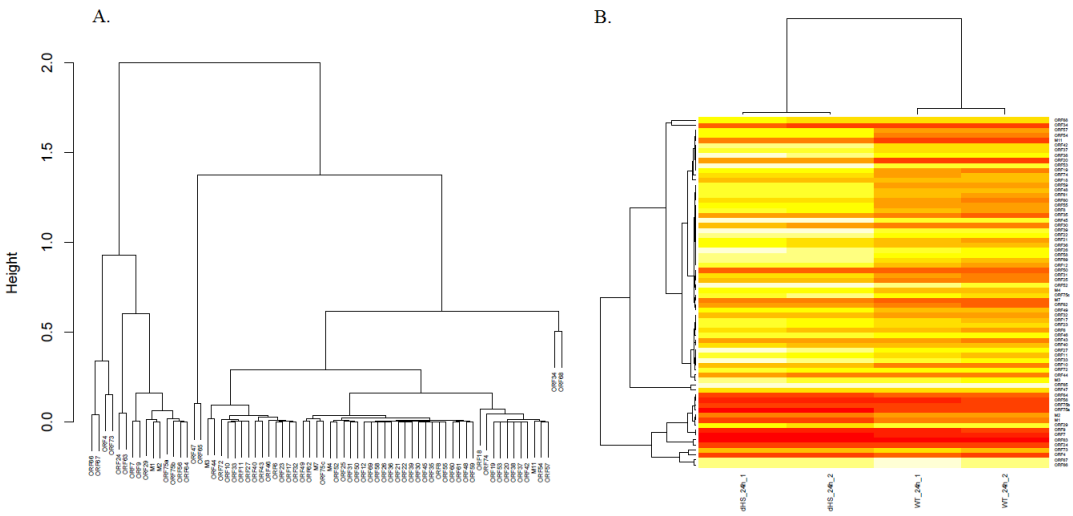

Figure 4.2: Hierarchical clustering of two different mutants of MHV68 viral ORFs (array 5) based on the log fold changes in expression. Figure-A. Cluster of ORFs. Figure-B. Heatmap of the two independent replicates of each virus infection.

from the analysis of the data-sets across the arrays, where the goal is to find stable and differentially expressed genes. The following section (sec. 4.3) describes our findings on the *gammaherpesvirus* induced host shutoff mechanism based on the expression changes for two different mutants of MHV68-infected viral and cellular genes (see [4] for complete write-up).

4.1 Data preparation and quality check[2]

Array data were derived from independent biological replicates at different infection conditions. The custom MHV68 tiled arrays in both 8 by 15,000 and 4 by 44,000 formats were designed using the Agilent eArray software. For both formats, a total of 11,940 tiled 60-mer oligonucleotides with 20-nucleotide (nt) spacing were generated on the basis of the MHV68 genomic sequence to enable triple probe coverage of every nucleotide of both strands of the virus [66]. RNA was labeled with linear amplification using an Agilent Quick Amp labeling kit. Initial reverse transcription used an oligo (dT) promoter-primer that enabled the generation of Cy5- or Cy3-labeled cRNA by T7 RNA polymerase. Adenovirus spike-in controls were added to each labeling reaction to allow normalization per the two-color spike-in kit instructions (Agilent).

[2]Experimental data prepared by: LTK Lab. See [65] for detail.

The Cy5-labeled reference RNA derived from an independent infection of NIH 3T3 fibroblasts at 8 h post-infection (hpi) with WT MHV68 was generated, purified, fragmented, and then hybridized in parallel with the Cy3-labeled sample RNA at $65°C$ for 17 h. Microarrays were scanned with 'Agilent Scanner Control' software, and hybridization signal intensities were quantified using 'Agilent Feature Extraction' software [65].

We had microarray data-sets at seven different experimental conditions of *Murine gammaherpesvirus 68* infection.

- *Array 1* – Represents microarray gene expression data at different time points (de novo time course experiment).
- *Array 2* – Examine the effect of drugs and of inhibiting NF-kB on infection.
- *Array 3* – Compares biological replicates and labeling techniques.
- *Array 4* – Examine changes of cellular genes over time and effect of inhibition of NF-kB on infection.
- *Array 5* – Examine changes in vSOX mutant (host shut-off defective) vs wild type (mutant rescue) viral and cellular genes at 24 hour time point.
- *Array 6* – Examine changes in 'tegument serine/threonine protein kinase' (ORF36) mutant vs wild type viral and cellular genes.
- *Array 7* – Examine changes in vSOX mutant vs wild type viral and cellular genes at 6 hour time point.

Raw data were processed by subtracting the median background signals from the mean signals in the Agilent feature extraction file and then normalized by multiplying log_2 values by the spike-in scale factor. We calculated a scaling factor from the linear fit of the spike-ins versus their concentration and scaled all the probe intensities with this scaling factor for each array. The expression value of each viral ORF was calculated as the median of the normalized log_2 green subtracted red signal of all tiling probes enclosed inside the ORF.

As a quality control measure, data from each replicate array were compared in box plots (box-and-whisker plot ignoring outliers, from R package '*graphics*'), density plots (R package '*stats*'), and scatter plot of wildtype vs wildtype variant and mutant vs mutant variant to measure consistency between array replicates. Figure. 4.1 shows the quality control plots of array 5 data. Array 5 data had two different experimental conditions (WT and ΔHS),

Table 4.1: Top 7 candidate housekeeping genes

Gene	μ	σ	$CV(\%)$	log_2MFC	$Max - 2 \times \sigma - \mu$
Fasn	11.834	0.395	3.342	1.561	-0.059
Cxxc1	11.394	0.395	3.467	1.482	-0.186
Gpi1	14.927	0.519	3.480	2.115	-0.376
Ube2i	14.435	0.521	3.609	2.094	-0.198
Polr2a	9.825	0.366	3.731	1.809	0.0043
Nedd8	13.691	0.522	3.816	2.100	-0.146
Ankrd10	9.718	0.382	3.935	1.622	-0.166

Table 4.2: Statistics of two commonly used candidate housekeeping genes

Gene	μ	σ	$CV(\%)$	log_2MFC	$Max - 2 \times \sigma - \mu$
Gapdh	13.856	0.735	5.310	3.216	-0.330
Hprt1	13.074	0.767	5.870	3.162	-0.194

with each having 2 biological replicates. Green signals show the original experimental conditions, red signals were used as reference. We linear fitted the data from the scatter plots using '*lm*' from R package '*stats*'; the black line at each scatter plot shows the fitted line through the data. The adjusted r^2 (coefficient of determination) value for the linear model fitted from wildtype vs wildtype data was 0.9622 and the corresponding r^2 value from the fit of mutant vs mutant data was 0.9095. Quality control plots verify that the replicates at each experimental condition matches well enough. Also, the reference signal is quite stable across the experimental conditions.

4.2 Data analysis across the arrays[3]

We analyzed each individual array data-set in different ways. At first we tried to group genes based on the changes in the expression. We clustered the ORFs from the array data to know which genes show similar behavior. Fig. 4.2A is showing the hierarchical clustering of the ORFs from array5 data (which contains two different array replicates - ΔHS and MR viruses) using the complete linkage method. The corresponding heatmap of the ORFs based on different experimental conditions is shown in fig. 4.2B. Here, the color changes from darker to lighter indicate increasing gene expression values.

Next, we sought to identify housekeeping genes, whose change in expression

[3]Partially completed (all results not published yet).

is relatively stable across the arrays. Generating a list of housekeeping genes is important, as these genes can be used as biological factors for normalization on the same set of genes in later experiments. To find the stable genes, we needed to combine the data across the arrays. We tried to merge data from six different arrays collected at different experimental conditions (excluding array7 - which had some important genes missing). At first we removed replicates from the array data that differed by more than 2-fold. Each array has expression data from duplicate genes and there are duplicate probe data for each gene. We merged array data first by probe name and then took the mean from each group. Next, we merged genes corresponding to different probe names but with the same gene name and again took the mean from each group. Then we quantile normalized the data (using R package LIMMA) to achieve consistency across the arrays, which forces the entire empirical distribution of each array to be identical.

A candidate housekeeping gene is defined as a gene with the most stable expression - having small coefficient of variation (CV) and a maximum fold change (MFC, the ratio of the maximum and minimum values observed within the dataset) below 2. A mean expression level lower than the maximum expression level subtracted with 2 standard deviation (σ) can also be used as a prerequisite for a candidate housekeeping gene [67]. To find the candidate housekeeping genes, we generated statistics of the merged quantile normalized data. We calculated standard deviation (σ) and coefficient of variation (CV) of expression for each gene across the arrays. However, no gene in the list fulfilled the criteria: $MFC < 2$. Based on our analysis, the top 7 probable candidate housekeeping genes are shown in table 4.1. Two commonly used candidate housekeeping genes statistics are shown in table 4.2. However, there is no universal set of rules that can be used to find housekeeping genes. In addition, viruses like MHV68 can have large effects on host transcription mechanism. As the data across the arrays had large variability due to the experimental conditions, it was difficult to identify stable genes. Relaxing some of the prerequisites in a careful way and exploring other conditions may help to better predict the candidate housekeeping genes. One possible way to verify the reliability of these putative housekeeping genes would be to normalize the data in different arrays using the mean change of expression of the stable genes and compare the values with the data normalized using other techniques (i.e. quantile normalization).

We separately analyzed microarray data on variants of MHV68 infection at different experimental conditions. To find the differentially expressed viral genes, we merged genes corresponding to the same ORF and took the median value from each group. Then the differentially expressed genes were identified

(from array5 and array7 data) by performing empirical Bayes moderated t-statistics using the Bioconductor LIMMA package (version 3.12.1) [68].

4.3 Gammaherpesviral gene expression and virion composition are broadly controlled by accelerated mRNA degradation[4,5]

Viruses use various ways to interact with host cell translation mechanism and to shutoff host gene expression. Part of the benefit could lie in reducing the competition for gene expression resources or to impair the immune response system inside the host cell. Earlier research works have shown for both alpha- and gammaherpesviruses that, host shutoff mutants exhibit defects in immune evasion, viral trafficking, and latency establishment [69, 70, 71].

Gammaherpesviruses promote host shutoff by inducing widespread cellular mRNA degradation [72, 73]. This viral subfamily includes Kaposi's sarcoma-associated herpesvirus (KSHV), Epstein-Barr virus (EBV), and the murine herpesvirus (MHV68). In KSHV, a special viral nuclease (SOX) is used to endonucleolytically cleave cytoplasmic mRNAs during lytic infection, leading to their degradation by the host exoribonuclease Xrn1 [74]. The phenotype of global mRNA degradation has been conserved in other SOX homologs in EBV (termed BGLF5) and MHV68 (termed muSOX) as well [73, 75]. An earlier study revealed that, MHV68 bearing muSOX mutant that is specifically defective for mRNA cleavage, resulted in defects in viral trafficking from the mouse lung to distal sites, and also resulted in a reduction of the viral loads during the time of peak latency establishment [69].

However, viruses generally have evolved a wide range of strategies to escape the effects of host shutoff. This phenomena have been observed for poliovirus, which inhibits cap-dependent translation by cleaving eIF4G, but enabling the viral mRNAs to translate [76, 77]. HSV-induced host shutoff involves inhibiting spliceosome assembly and blocking the biogenesis of host mRNAs [78]. However HSV mRNAs being largely unspliced, can successfully overcome the blocking stage [79, 80]. SARS coronavirus acts in host shutoff by cleaving cellular mRNAs of the host, but its viral mRNAs bear a protective 59 leader sequence, which aids in protecting the viral mRNAs from cleavage [81].

Based on the results from earlier studies performed on various virus life

[4]E. Abernathy, K. Clyde, R. Yeasmin, L. T. Krug, A. Burlingame, L. Coscoy and B. Glaunsinger. Gammaherpesviral gene expression and virion composition are broadly controlled by accelerated mRNA degradation. PLoS Pathog, 2014

[5]RT-qPCR analysis and mRNA half-life analysis results are contributed by BG Lab.

cycles, the general assumption is that, viral transcripts possess some strategy to escape the shutoff mechanism. SOX and muSOX are expressed with early gene kinetics beginning at $8-10$ hours post infection (hpi) and continues through the end of the viral lifecycle [73]. Hence, it has been assumed for SOX and muSOX-induced viruses as well that, viral gene expression remains unaffected by the host shutoff activity, though there was no direct evidence of such phenomena. However, our analysis revealed that, viral gene expression is widely affected by the muSOX-induced RNA degradation during MHV68 infection. A majority of viral mRNAs are reduced during lytic infection in the WT infected cells compared to cells infected with the vSOX mutant virus, and the escapee population is enriched with viral non-coding RNAs.

4.3.1 Expression of viral mRNAs are largely degraded during host shutoff

Gammaherpesvirus uses SOX protein to induce widespread degradation of cellular mRNAs [72, 73]. The impact of this protein on viral mRNAs are yet to explore. We used two different MHV68 mutants to address this question - one that is host shutoff defective (ΔHS) and another one is mutant rescue virus (MR) in which mutation was restored to wild type (array 5 data in sec. 4.1).

The ORF37 gene in ΔHS (which encodes the SOX homolog, MHV68 muSOX) contains an R443I mutation that causes muSOX to be selectively defective for mRNA degradation [69]. Viral transcript abundance was comprehensively evaluated using an MHV68 microarray platform containing 12,000 tiled 60-mer probes that provides 3-fold coverage of each strand of the viral genome. Relative transcript levels were measured in NIH 3T3 cells infected at an MOI of 5 at 24 hours post infection (hpi), by which time the phenotype causing mRNA degradation is well established. Surprisingly, the majority of viral mRNAs from all three kinetic classes (immediate early (IE), early (E), and late (L)) were significantly down-regulated during the MR infection as compared to the ΔHS infection (fig. 4.3A, red bars). This phenomena suggests that, viral transcripts do not escape the muSOX-induced degradation.

RT-qPCR was used as an independent measure of viral mRNA levels during infection for three representative genes (ORFs 8, 49, and 54) after normalization to the host shutoff resistant 18S ribosomal RNA, which confirmed the earlier finding [74, 75]. However, some of the genes (ORFs 4, 9, 65, 68, and 73) found to be decreased in level during MR infection compared to the ΔHS in the qPCR analysis, which appeared to remain unchanged based on the microarray data analysis (fig. 4.3B). This could be because of microarray data underestimating the extent to which muSOX-induced infection affects the viral

mRNAs. RT-qPCR analysis of these transcripts in different cell types yielded similar results ([4], S2A-B).

To figure out whether the degradation of viral transcripts are the direct consequence of muSOX-induced activity or due to the degradation of cellular proteins, half-life of representative IE (ORF57), E (ORFs 54 and 55), and L (ORF8) viral transcripts (following the infection of 3T3 cells with MR or ΔHS virus) were compared. In the analysis, significant increase in the transcript levels were observed for ΔHS mutation (fig. 4.3C-F). These results suggest that, viral mRNAs are specifically targeted for degradation during gammaherpesvirus infection. Hence, the host cell degradation mechanism imposed by muSOX is not limited to cellular mRNAs only, it affect viral mRNAs as well.

4.3.2 Escapee population is enriched with non-coding RNAs

Although the majority of viral RNAs seem to be targeted for degradation during the host shutoff process of MHV68 vSOX, a subset of the genes escape the degradation, which are shown by the green bars in fig. 4.3A. In the escapee population, M1 and M2 mRNA levels are higher in MR infected cells compared to ΔHS infected cells. RT-qPCR analysis revealed similar results [4]. Among other escapees in the population, non-coding RNAs (ncRNAs) are highly enriched. Analysis of viral vs non-coding RNA distribution revealed that, most of the viral coding mRNAs are affected by the muSOX-induced infection, while the escapee population is widely enriched for the ncRNAs [4]. RT-qPCR analysis confirmed that, the levels of the non-coding RNA subset, the viral tRNA-like RNAs, were not decreased during the MR infection relative to the infection with the ΔHS virus. Several long non-coding RNAs termed EGRs (Expressed Genomic Regions) also escaped the down-regulation, which was confirmed by both microarray data and qPCR analysis (fig. 4.3A). However, some of the EGR levels (24, 27, and 29) were decreased during the MR infection relative to the ΔHS MHV68 infection [4]. The functionality of EGRs are not fully characterized yet. EGRs, down-regulated during the infection possess features of mRNAs such as polyA tails. All of these findings suggest that, viral mRNAs are preferentially targeted by the muSOX during the host shutoff process, but the non-coding RNAs broadly escape this degradation.

4.3.3 Discussion

Altogether the analysis outcome reveals that, along with targeting for cellular mRNAs, majority of viral mRNAs are also targeted for degradation during

lytic MHV68 infection, which selectively spares non-coding RNAs and some expressed genomic regions. Further analysis revealed that, degradation of viral mRNA level is particularly important for the regulation of viral loads at later stages of the viral life cycle (see [4] for detail). This is in contrast to the earlier findings of host shutoff inducing viruses, which possess specific strategies to escape the degradation.

Earlier studies showed that, SOX homologs preferentially target for RNAs transcribed by Pol II, but fails to degrade RNAs transcribed by Pol I and III [74, 75]. In our analysis, we also noticed viral mRNAs are preferentially targeted over non-coding RNAs and EGRs. However, some EGRs were targeted by muSOX for degradation, further studies may reveal unknown functionalities of these expressed genomic regions. Earlier analysis showed that, some of the viral and cellular transcripts escaped the degradation process [72, 82], it may be because of lacking some SOX-targeting features or these mRNAs may contain some specific protective features against muSOX activity [83]. Further study on these mRNAs could be interesting as well.

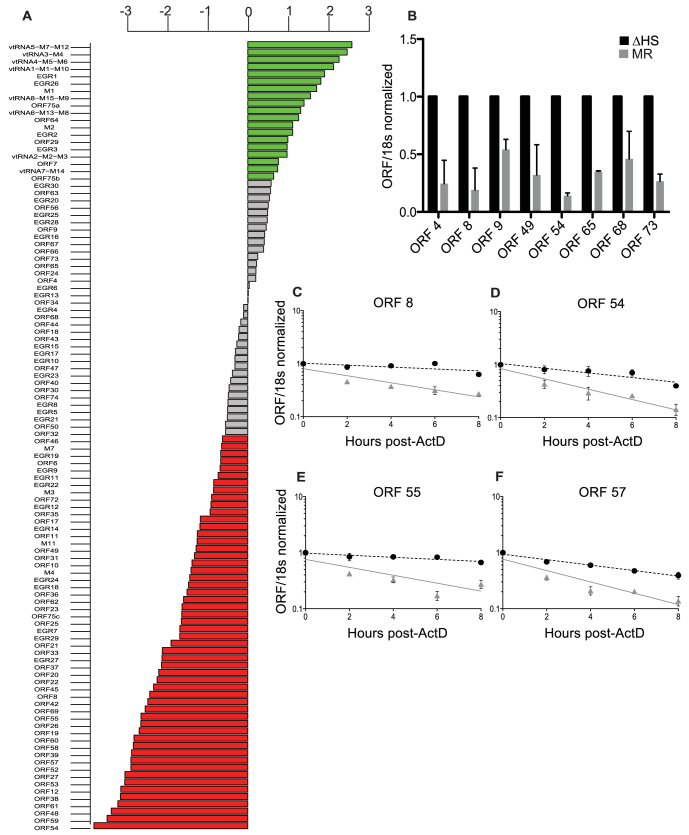

Figure 4.3: Expression of the majority of viral mRNAs is dampened during host shutoff. (A) For viral genes, the log_2 fold change in expression upon infection with WT compared to ΔHS MHV68 expression was plotted and data points were colored to indicate the adjusted P values, with green points indicating positive log_2 fold change with p-value < 0.05 and red indicating negative log_2 fold change with p-value < 0.05. (B) RT-qPCR was used to validate selective viral transcripts. RNA was isolated at 24 hpi from NIH 3T3 cells infected with MR or ΔHS MHV68 at an MOI of 5. Transcript levels were normalized to 18s and ΔHS levels set to 1. (C-F) mRNA half-life analyses were conducted by infecting NIH 3T3 cells with MR or ΔHS MHV68 at an MOI of 5. At 18 hpi, $2\mu g$ of Actinomycin D was added to block transcription and RNA was harvested at the indicated times thereafter. RT-qPCR was performed with ORF-specific primers and probes to determine mRNA levels. The black dotted line indicates the best-fit line for the ΔHS virus, and the grey solid line indicates the best-fit line for the MR virus.

Chapter 5

Measurement of average decoding rates of the 61 sense codons in vivo[1,2]

Most of the amino acids are encoded by more than one codons. The frequency of usage of these codons are not equal. The reason or significance behind the difference in codon usage frequency is not clear yet. One compelling hypothesis is that, codons that are translated faster are more commonly used. Relative rates of protein production, protein folding, ribosome traffic-management etc. can be explained by the hypothesis. However, there is little direct, in vivo evidence regarding codon-specific translation rates. We have analyzed high-coverage ribosome profile data from yeast using a novel algorithm and have deduced events at the A and P-sites of the ribosome. Codons are decoded at varying rates in the A-site. In general, frequently used codons are decoded more quickly than rare codons, and AT-rich codons are decoded more quickly than GC-rich codons. At the P-site, we see that *proline* is slow in forming peptide bonds. We also have applied our algorithm to short footprints (which captures a different conformation of an active ribosome), and found strong, amino-acid specific effects independent of codon specific preferences, that may reflect interactions with the exit tunnel of the ribosome.

Different synonymous codons are used in genes at very different frequencies, and the reasons for this biased codon usage have been debated for three decades [51, 88, 89, 90, 91, 92, 93, 94]. In particular, it has been suggested that the frequently-used codons are translated more rapidly than rarely-used codons,

[1] J. Gardin, R. Yeasmin, A. Yurovsky, Y. Cai, S. Skiena and B. Futcher. Measurement of average decoding rates of the 61 sense codons in vivo. eLife, 2014

[2] Direct excerpts from [84].

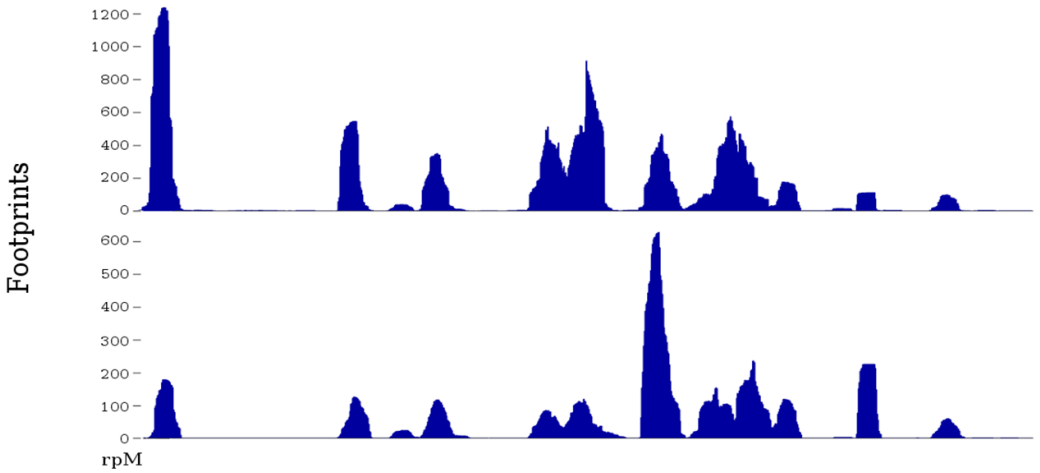

Figure 5.1: Ribosome profiles of the TDH1 gene from two independent experiments. Top profile is from the data collected from Ingolia et al. [85]; bottom profile is from the SC-lys dataset (sec. 5.5).

perhaps because tRNAs for the frequent codons are relatively highly expressed [8]. However, there have also been competing hypotheses, including the idea that frequently-used codons are translated more accurately [8]. Genes are often recoded to use frequent codons to increase protein expression [95, 96], but without any solid understanding of why this manipulation is effective. There is little or no direct in vivo evidence as to whether the more common codons are indeed translated more rapidly than the rarer codons. Even if they are, the fact that translation is typically limited by initiation, not elongation, leaves the effectiveness of codon optimization a puzzle [8].

Ribosome profiling [85] allows observation of positions of ribosomes on translating cellular mRNAs. The basis of the method is that a translating ribosome protects a region of mRNA from nuclease digestion, generating a 30 base *footprint*. The footprint is roughly centered on the A-site of the ribosome. If some particular codon in the A-site were translated slowly, then the ribosome would dwell at this position, and so footprints generated from ribosomes at this position would be relatively common. Thus, if one looked at the number of ribosome footprints generated along an mRNA, there should be more footprints centered at every codon that is translated slowly, and fewer centered at every codon translated rapidly; in principle, this is a method for measuring rates of translation of individual codons.

Experimentally, there is dramatic variation in the number of footprints generated at different positions along any particular mRNA [97] (fig. 5.1). However, these large peaks and valleys do not correlate with particular codons [97, 98]. It is still unclear what features of the mRNA cause the peaks and

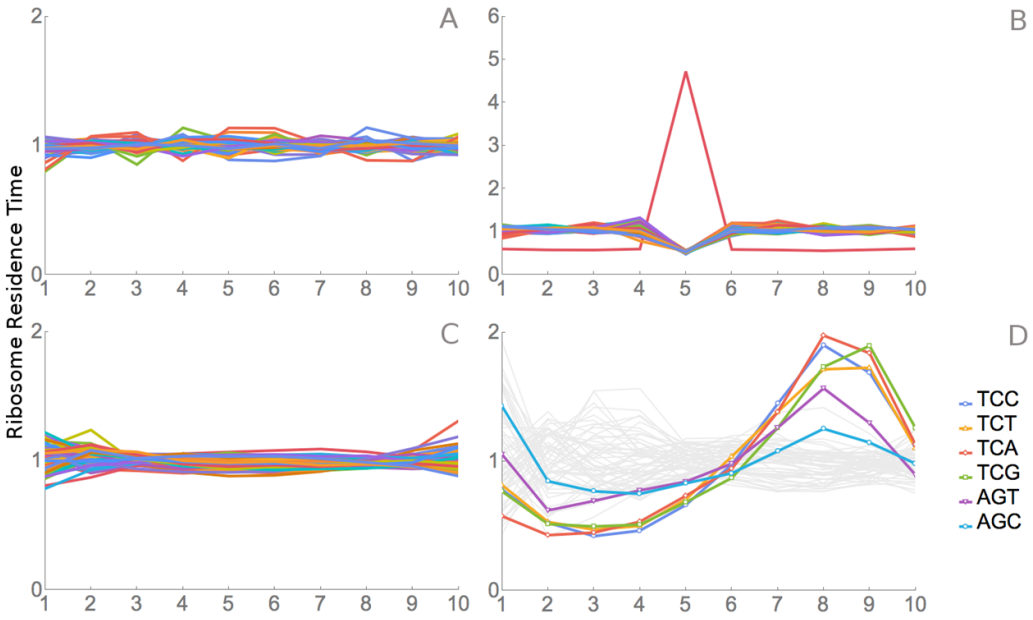

Figure 5.2: Validation for ribosome residence time analysis. (A) Real footprint data from the SC-lys dataset were randomly assigned to codons. No signal is detected from this randomly generated data-set. (B) A dataset of 2 million simulated reads was generated but biased to give more reads over the codon AAA at position 5. Sharp peak at position 5 is noticed for AAA. (C) RNA-seq data processed in a way similar to ribosome profiling, our algorithm does not detect any signal for this data-set. (D) Real ribosome footprint data from Li et al. [86] were analyzed. Here, E. coli were starved for *Serine*. Among the *Ser* codons, highest peak is detected for the rarest codon *TCA*, and the lowest peak is noticed for *AGC*, which is the most common *Ser* codon in E.coli.

valleys, though there is evidence that *prolines*, or a poly-basic amino acid stretch, contribute to a slowing of the ribosome and a peak of ribosome footprints [97, 98, 99].

Still, the fact that *prolines* and poly-basic amino acid stretches affect translation speed does not tell us whether different synonymous codons may also cause smaller effects. This question was investigated by Qian et al. [57] and Charnesky and Hurst [98] using the yeast ribosome profiling data of Ingolia et al. [85]. Neither group found any effect of different synonymous codons on translation rate - that is, perhaps surprisingly, each codon, rare or common, appeared to be translated at the same rate [57, 98].

We have re-investigated this issue with two differences from these previous investigations. First, we have generated four yeast ribosome profiling data-sets by optimized methods, including the flash-freezing of growing cells before the

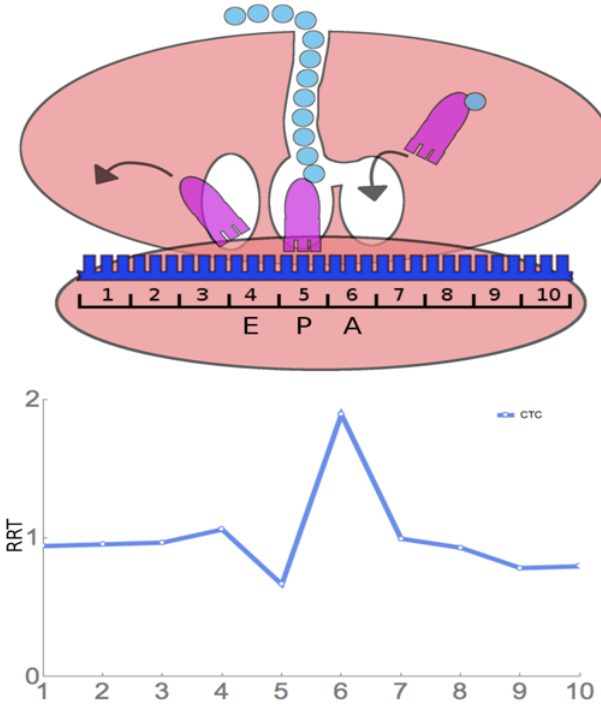

Figure 5.3: Principle of ribosome residence time analysis. The ribosome protects a 30 nt *footprint* of RNA centered around the A, P, and E sites (positions 6, 5, and 4). Position 6 is likely the ribosome A-site (where the rare *Leu* codon *CTC* shows a peak).

addition of cycloheximide (5.5); Ingolia et al. added cycloheximide before harvesting cells. Second, we have developed a novel method of analysis, designed with the knowledge that, at best, codon decoding rates could account for only a small portion of the variation in ribosome footprints across an mRNA (5.5). The combination of optimized data and novel analysis reveals that different codons are decoded at different rates.

5.1 Previous works

A well-known method for assessing the translation status of mRNAs is polysome profiling, which involves microarray data analysis to measure the mRNA fragments containing varying number of ribosomes [100, 101, 102]. In polysome profiling, velocity sedimentation on a sucrose gradient is used to separate mRNAs into groups based on the number of ribosomes attached to it and the resultant data is analyzed using quantitative microarray analysis [101]. Polysome profiling can be used as an alternative method for ribosome profiling, in particular to perform mechanistic studies of translational control. However, ribo-

some profiling avoids the difficulty in resolving the exact number of ribosomes attached to a highly ribosome-loaded transcript and hence provides more precise expression measurement [100]. Moreover, exact ribosome position in an mRNA cannot be detected in polysome profiling [103]. However one advantage of polysome profiling is that, it provides a way to study the translation status of the entire transcript.

Ribosome profiling data was first generated by Ingolia et al. [97, 100]. The process starts with cell lysis, where in vivo positions of the translating ribosomes are kept intact. Next, mRNA molecules bound to ribosomes are isolated and nuclease digestion is used to remove unprotected mRNA. Recovered footprints are purified and ligated to a single-stranded linker to serve as a priming site for reverse transcription. Sequences are then amplified in strand-specific manner using PCR amplification. One problem with ribosome profiling data is that, de-convolving repetitive sequences or alternative transcripts in these data is quite hard, as the ribosome-protected mRNA regions are quite short [103].

Several researchers worked on the data-set generated by Ingolia et al. Tuller et al. suggested that, the speed of the ribosome tends to be slower at the beginning of the coding sequence [104]. In the original data-set, a relatively large peak of ribosome footprints was detected immediately after the initiator AUG, which was interpreted as being due to slowly-translating codons at this position. However, the data-set was generated by adding cycloheximide to growing cells, and an alternative explanation to the finding can be, cycloheximide can only slow down or freeze elongating ribosomes, but it does not stop the addition of new ribosomes at the beginning of the gene. This phenomena would also cause a peak of footprints after the initiator AUG. Recent ribosome profile data-sets, where cells are flash-frozen before the addition of cycloheximide, do not display the large peak in ribosome footprints downstream of the initiator codon, consistent with this alternative explanation.

5.2 Results

In principle, using the ribosome footprint data to establish occupancy as a function of position might seem easy: align the reads to the reference genome to identify the ten or so codons under each read, and tabulate the frequency of each codon observed in each position. Analysis of this general kind has been carried out previously, but without detecting codon-specific differences in decoding rates [57, 98]. However, this analysis in its simplest form would overweight the highly expressed genes, which account for a large fraction of total reads - that is, a relatively small number of highly-expressed genes would

dominate the analysis. But because there are extreme peaks and valleys in ribosome footprint profiles (fig. 5.1), and because these are not primarily due to codon usage, this simple analysis would likely fail, because the results would depend mainly on a relatively small number of chromosome positions, and because of the peak-to-valley variability affecting these positions. Defining the right normalization to compensate for differences in gene expression, gene length, sequence composition, etc., is complicated and problematic.

Instead, we have opted for a simpler approach. We independently analyze many selected regions (windows) where the effects of codon usage are particularly easy to assay. For each codon, we identify all translated regions in the genome where a particular codon (say CTC) occurs uniquely within a window of ten codons upstream and ten codons downstream - that is, a window 19 codons wide, with the codon of interest occurring exactly once, at position 10 of the 19-position window. For footprints 10 codons long, there are exactly ten classes of footprints that contain this particular CTC, and fit entirely in the window. That is, the CTC of interest can occur at position 1 of the footprint, or position 2, ..., or position 10. Analysis was restricted to windows with at least 20 total reads and at least 3 non-empty classes. For our four data-sets discussed below, there were an average of 408, 1586, 1749, or 2868 qualifying windows per codon, respectively (more windows for the abundant codons, fewer for the rare codons).

In the absence of any codon preference of the ribosome, there should be a uniform distribution of footprints across the ten positions. That is, in a window centered on CTC and containing 100 footprints, one expects 10 footprints at each of the 10 positions, a relative frequency of 0.1 (10/100) at each position. On the other hand, if the ribosome were to dwell for an extended time over the CTC whenever that codon was at, say, position six of the footprint, then there might be 30 footprints with CTC in position six, and about 8 footprints at each of the other nine positions, thus giving a frequency distribution with a peak at position 6. Many such relative frequency distributions can be fairly averaged over all windows over all genes centered on a specific codon. Regions on highly expressed genes can be fairly compared with similar regions on genes with lower expression, because we are dealing with relative frequency distributions. Each window thus represents an independent trial of the ribosome's dwell time over each given codon. Averaging over the hundreds or thousands of windows in the genome generates a statistically rigorous analysis. Note that we do not attempt any normalization based on gene expression - instead, we take each qualifying window as an independent experiment, regardless of level of expression, then average all frequency distributions from all windows for each codon. A related idea was also used by Lareau et al. [87], although on significantly different

Table 5.1: Top ten RRT at position 8 in E. coli starved for *Serine*

Codon	*AA*	*Usage*	*RRT*
TCA	Ser	8.1	1.98
TCC	Ser	9.0	1.90
TCG	Ser	8.8	1.73
TCT	Ser	8.7	1.71
AGT	Ser	9.4	1.57
ATA	Ile	5.5	1.42
AGC	Ser	16.0	1.25
ATT	Ile	29.7	1.18
CCT	Pro	7.2	1.15
CCA	Pro	8.4	1.13

data, and with normalization by gene.

The relative frequency averaged over all windows is a number between 0 and 1, and we compare this to the baseline frequency (0.1) (total footprints over 10 positions) to compute a final statistic, which we call the *Ribosome Residence Time*, or RRT. For instance, if the average relative frequency for a codon at a particular position is 0.1, then the RRT is 1, and we interpret this to mean that the ribosome spends the average amount of time at the given codon at the given position. An RRT of 2 suggests the ribosome spends twice as long as average at the given codon.

5.3 Validation of ribosome residence time analysis

We tested this method of analysis using simulated and real positive and negative control data. For a simulated negative control, we assigned real footprint data from our SC-lys data-set to random codons, and did RRT analysis. As expected, all codons at all positions show an RRT of about 1, i.e., no signal (fig. 5.2A). For a simulated positive control, we generated a simulated data set of 2 million 10-codon reads over coding genes, but we biased these simulated reads to give more reads for the codon AAA at position 5 of the footprint. As expected, RRT analysis shows a peak for AAA at position 5 (fig. 5.2B).

For a real-data negative control, we pooled the control mRNA-seq data for 30bp fragments from our four experiments (sec. 5.5), and analyzed these mRNA fragments. Since this RNA came from a total naked RNA preparation, there were no ribosomes and no ribosome footprints, so there should not be

any signal from translation, even though we are analyzing real 30*bp* RNA fragments. Indeed, RRT analysis shows no peak at positions 2 through 9 of these fragments (fig. 5.2C). However, there are modest deviations from 1 at the termini, positions 1 and 10. We attribute these to some base-specificity for the enzymatic reactions used to generate the fragment library [105, 106, 107]. Supporting this interpretation, the same peaks and valleys at positions 1 and 10 (i.e., the same base-specificity) were seen in real ribosome-footprint data (see below).

For a real-data positive control experiment, we used the *E. coli* data generated by Li et al., who starved E. coli for serine, and did ribosome profiling [86]. Because of the starvation for serine, there is an expectation that all six serine codons should be decoded slowly, and so should have high RRT values. This proved to be the case (fig. 5.2D). The six serine codons had 6 of the 7 highest RRT values at position 8 (fig. 5.2D, table 5.1), which presumably represents the A-site in this experiment. Note that because these are E. coli ribosomes, the phase of the footprint (i.e., the position of the A-site in the footprint) is different from its phase with regard to yeast ribosomes (see below). The RRT analysis of *E. coli* footprints also showed interesting variation at positions 2, 3, and 4 (fig. 5.2D), which we will consider elsewhere.

Lareau et al. [87] starved S. cerevisiae for histidine using the *His3* inhibitor 3-aminotriazole. This was another potential positive control, where the two *His* codons should be decoded slowly. We analyzed these ribosome profiling data. However, of the 11 million reads obtained in that experiment, about 10.6 million mapped to ribosomal RNA. The remaining $\sim$ 0.4 million reads mapped to mRNA, but gave only 10 (ten) total windows passing our quality filters for RRT analysis, and this is too few. However, when we relaxed the filters to obtain more (albeit lower quality) windows, we observed obvious peaks (high RRT values) for both histidine codons at position 6 specifically in the 3-aminotriazole experiment (data not shown).

5.3.1 Ribosome residence time analysis of codons

Having found that RRT analysis gives the expected results in control experiments, we applied it to the analysis of four of our ribosome profiling experiments. Our experiments differ from those of Ingolia et al. and Lareau et al., in that in those studies, cycloheximide was added to the growing yeast culture before harvesting [85, 87], whereas we harvest by flash-freezing, and later add cycloheximide to the frozen cells (sec. 5.5). The nature of our results is shown in fig. 5.3 using the rare *Leu* codon *CTC* as an example. In this example, 10 codon (30 nucleotide) footprints that have *CTC* as the first codon have about the average relative frequency - that is, they have about the same relative

frequency as footprints with any other codon at the first position. Similarly when *CTC* is in the 2nd, 3rd, 4th, 7th, 8th, 9th, and 10th positions. However, there is a relative over-abundance of footprints that have *CTC* at the 6th position. In fact, for *CTC* at the 6th position, averaged over 451 windows (in the case of this rare codon), there are 1.89-fold more footprints than at the baseline. This suggests that ribosomes move relatively slowly when *CTC* is at the 6th position, and, therefore, these ribosomes are more frequently captured as footprints. We say that *CTC* has a Ribosome Residence Time (RRT) of 1.89 at position 6.

Fig. 5.4 shows data for all 61 sense codons from one of four experiments, the *SC-lys* experiment. In a large majority of cases, a codon has its highest or lowest footprint abundance when the codon is in position 6. We interpret this to mean that the codon affects the rate of ribosome movement when the codon is in position 6, which we believe to be the A-site of the ribosome (see below for further support for this assignment). The behavior of the six *Leu* codons and the four *Thr* codons, is highlighted in fig. 5.4B and 5.4C. Footprint frequencies also differ from the average in a specific way at positions 5 (fig. 5.4D) (see below) and 1 and 10, the two ends of the footprint. We attribute variation at positions 1 and 10 to some base-specificity for the enzymatic reactions involved in generating and analyzing ribosome footprints [105, 106, 107]; the same variations are seen in reactions with naked RNA fragments.

Fig. 5.5A shows the deduced rate of ribosome movement for each codon, plotted against the frequency of codon usage. There is a good correlation ($r = -0.52$); that is, the ribosome moves faster over the more common codons. There is also a correlation, albeit weaker, with the AT-richness of the codon. *AT-rich* codons are decoded somewhat faster than average, while *GC-rich* codons are decoded more slowly (fig. 5.5B). The mean RRT of codons with 3 or 2 *GC* residues was 1.23, while the mean RRT of codons with 1 or 0 *GC* residues was 1.01, a statistically significant difference ($p < 0.003$ by a two-tailed t-test).

Table 5.2 shows the Ribosome Residence Time at position 6 for each of the 61 sense codons. The slowest codon is the rare *Leu* codon *CTC*. Relatively, the ribosome spends about 1.9 times as long with a *CTC* codon in the A-site as it does at the average codon. If the yeast ribosome spends 50 milliseconds [108] on an average codon in the A-site, then the RRT suggests it spends about 95 milliseconds on *CTC* codons. The fastest codon is the relatively abundant *Thr* codon *ACC* (fig. 5.4C, table 5.2), where it spends 0.70 times as long as average (i.e., about 35 milliseconds).

There are also peaks at position 5 (fig. 5.4A, 5.4D), which we interpret as the ribosome's P-site, where the peptide bond is formed. All four *Pro* codons

are high at position 5: *CCT*, *CCA*, and *CCC* are the three slowest codons at position 5, while *CCG* is 6th (fig. 5.4D, table 5.3). *Proline* is a unique amino acid in having a secondary rather than a primary amino group, and so is less reactive in peptide bond formation. *Proline* forms peptide bonds slowly [109, 110, 111], and it has been associated with slow translation in footprinting experiments [97]. Our result that the ribosome slows with *proline* at position 5 is consistent with this, and tends to confirm our assignment of position 5 to the P-site and, therefore, position 6 to the A-site. A few other residues also seem slightly slow at position 5 (e.g., *Asn*, *Gly*, see table 5.3 and appendix A.1), possibly due to low reactivity in peptide bond formation [109].

All four *proline* codons also have high RRT at position 6, the A-site (fig. 5.4D, table 5.2). The dipeptide *ProPro* is translated very slowly [112, 113, 114]. We wondered whether the apparent slowness of *proline* at both positions 5 and 6 was an informatic artefact due to extreme slowness for *Pro-Pro* dipeptides. We redid the original analysis after excluding all footprints encoding *ProPro* dipeptides. Results did not change significantly; *Pro* still appeared to be slow at both positions 5 and 6 (fig. 5.6A). On the other hand, when we looked specifically at footprints containing a *ProPro* dipeptide, there was a very large peak at position 5 (fig. 5.6B), consistent with the very slow peptide bond formation seen in studies cited above.

To establish repeatability, we generated and analyzed three other ribosome profiling data-sets and also re-analyzed previously-published data [85]. All five data sets gave qualitatively similar results; pairwise correlations for RRT at position 6 ranged from 0.22 to 0.96 between the data-sets (table 5.4). The poorest correlation (0.22) was a correlation with the previously-published data-set, which was generated using significantly different methods than our data-sets. In particular, that data-set was generated by adding cycloheximide to the growing culture, then harvesting [85], whereas our data was generated by flash-freezing first, then adding cycloheximide to the frozen cells. Complete results for all five experiments are given in appendix A.1 (table A.1, A.2, A.3, A.4 and A.5). More recently, we also subjected the long footprint data of Lareau et al. [87] to RRT analysis, and obtained correlations at position 6 of 0.21, 0.47, 0.23, and 0.27, respectively, for their *untreated 1*, *untreated 2*, *untreated merge*, and *cycloheximide 1* experiments to our SC-lys experiment. Again, these experiments were carried out in a significantly different way from ours and it is not surprising that the correlations are modest. It is reassuring that a positive correlation can be seen even for experiments where no cycloheximide was used.

There are strong correlations between codon usage, the number of tRNA genes for the relevant tRNA, and tRNA abundance [6, 50, 54, 115]. Although

one cannot determine causation from this correlation [8], nevertheless it is consistent with idea that the rate of decoding in translation is at least partly limited by tRNA concentration. Most of our results are consistent with this. However, there are some interesting exceptions. In yeast, the 61 sense codons are decoded by only 42 tRNAs. There are 12 pairs of codons that share a single tRNA (e.g., *Phe* TTC and TTT, *Tyr* TAT and TAC, etc.) [116]. In many but not all cases, the RRT of the two codons is similar (table 5.2), consistent with the *concentration* hypothesis. However, there are also cases where the RRT appears to be significantly different for two codons sharing the same tRNA. For instance, the *Cys* codon TGC has an RRT of 1.23, while TGT has an RRT of 0.81 (table 5.2). Both codons are recognized by the same tRNA, which in this case is complementary for TGC, and wobble for TGT. Similarly, the *Gly* codon GGC has an RRT of 1.22 (tRNA is complementary), while GGT has an RRT of 0.93 (tRNA is wobble). Both these relationships ($RRT_{TGC} > RRT_{TGT}$, and $RRT_{GGC} > RRT_{GGT}$) were true in all five data-sets (appendix A.1). In both cases, the perfect match is decoded more slowly than the wobble match and in both cases, the slower, complementary pairing has a G:C match at the third (i.e., wobble) position. These and other similar examples (not shown) suggest that the RRT depends on more than just the concentration of the relevant tRNA. Perhaps the long RRT for these GC-rich codons is related to the time needed to eject incorrectly paired anticodons of incorrect tRNAs, although this explanation is somewhat at odds with the literature [117, 118]. Alternatively, it has been suggested that translocation can occur more quickly when the codon-anticodon interaction is weaker [119, 120].

5.3.2 RRT analysis of short footprints

Recently, Lareau et al. made the exciting discovery that ribosome profiling on cells that have not been treated with any drug yields two classes of footprints, long (28-30 nucleotides) and short (20-22 nucleotides) [87]. It is the long class that is seen in cycloheximide experiments, and which we have characterized above. The short (20-22 nt) footprints seem to represent a different conformation of the ribosome, perhaps one that occurs when the ribosome translocates along the mRNA. Furthermore, Lareau et al. found that treatment of cells with the elongation inhibitor anisomycin efficiently generates short footprints. Lareau et al. suggest that the long and short footprints are reporting on two different states of translation [87].

We applied RRT analysis to the short footprints generated by Lareau et al., with special focus on the footprints after anisomycin treatment. All three of their anisomycin data-sets were studied, and the pairwise correlations between the RRT results for these three data-sets were very high, ranging from 0.89 to

0.998. Partial results are shown in fig. 5.7 and table 5.5 and complete results are shown in appendix A.1 (A.6, A.7 and A.8). RRT analysis showed a series of peaks at different positions along the 7-codon footprint. The RRT values for the short footprints did not significantly correlate with RRT values for the long footprints, even when the phases of the footprints were shifted. This suggests, in agreement with Lareau et al., that the short and long footprints are indeed reporting on different translational processes. Furthermore, for the short footprints the RRT values are amino acid specific, while for the long footprints at position 6, the RRT values are codon specific (table 5.2; table 5.5; fig. 5.4, fig. 5.7, fig. 5.8). This again indicates that the two kinds of footprints are reporting on different translational processes. The amino acids in the peaks at positions 3, 5, and 6 are shown in table 5.5: the peak at position 3 contains glycine; the peak at position 5 contains smallish hydrophobic amino acids (*Leu*, *Val*, *Ile*, and to some extent *Phe*), and the peak at position 6 is dominated by the two basic amino acids, *Arg* and *Lys*. It has previously been shown that basic amino acids can cause a pause in elongation by interacting with the ribosome exit tunnel [98, 99, 121]. The basis of the anisomycin arrest is partly but not fully understood [122, 123], and so it is difficult to clearly interpret these results (but see sec. 5.4). Nevertheless, the application of RRT analysis to the anisomycin-generated footprints gives strong, specific signals that are unlikely to be explained by a random process. We note, however, that results from the short footprints from untreated (no anisomycin) cells are only modestly correlated (0.23) with results from short footprints from the anisomycin-treated cells (data not shown).

It appeared that the RRT values at position 6 for the long footprints were codon-specific (fig. 5.4, table 5.2), while the RRT values for the short footprints were amino-acid specific (fig. 5.7, table 5.5). To confirm this, we developed a statistical test for the coherence of the results for a particular amino acid (sec. 5.5). Briefly, this method tests whether every codon for a particular amino acid behaves similarly, and yields a small p-value if it does. Indeed, this analysis confirms that the short footprints give results specific to the amino acid, while the long footprints generally do not (i.e., the long footprints are codon-specific) (fig. 5.8). This suggests that the long footprints are reporting on the process of decoding (which depends on specific codons), while the short footprints are reporting on events after decoding.

5.4 Discussion

To our knowledge, this is the first measurement of the differential rate of translation of all 61 codons in vivo. There is a correlation between a high

Table 5.2: Ribosome residence rime at position 6. Usage of each codon per 1000 codons and the ribosome residence time (RRT) at position 6 (the A-site of the ribosome). The p-value for a difference between the calculated RRT value and an RRT value of 1 is shown. P-values less than or equal to 0.001 are marked with an asterisk.)

Codon	AA	Usage	RRT	p-value	Codon	AA	Usage	RRT	p-value
CTC	Leu	5.4	1.892	*0.0001	TTT	Phe	26.1	1.0483	0.0529
CCC	Pro	6.8	1.7148	*0.0001	GAA	Glu	45.6	1.0405	0.0538
GGG	Gly	6	1.6089	*0.0001	AGA	Arg	21.3	1.0132	0.3014
AGG	Arg	9.2	1.5948	*0.0001	TTC	Phe	18.4	1.0001	0.4955
ATA	Ile	17.8	1.5667	*0.0001	GCG	Ala	6.2	0.9949	0.465
GGA	Gly	10.9	1.5582	*0.0001	TCC	Ser	14.2	0.9892	0.3341
TGG	Trp	10.4	1.5257	*0.0001	TTA	Leu	26.2	0.9853	0.3166
GTG	Val	10.8	1.5194	*0.0001	TCT	Ser	23.5	0.9813	0.2249
CGC	Arg	2.6	1.4532	*0.0001	CAT	His	13.6	0.9302	0.0188
CGA	Arg	3	1.447	*0.0008	GGT	Gly	23.9	0.9257	*0.0003
CGG	Arg	1.7	1.436	*0.0010	ATG	Met	20.9	0.923	0.0027
TCG	Ser	8.6	1.4273	*0.0001	ATT	Ile	30.1	0.922	*0.0005
CCA	Pro	18.3	1.3817	*0.0001	TTG	Leu	27.2	0.9202	*0.0001
ACA	Thr	17.8	1.347	*0.0001	CTG	Leu	10.5	0.916	0.0139
CCG	Pro	5.3	1.3124	*0.0001	AAT	Asn	35.7	0.8785	*0.0001
GTA	Val	11.8	1.3055	*0.0001	AAA	Lys	41.9	0.8781	*0.0003
GCA	Ala	16.2	1.2847	*0.0001	CGT	Arg	6.4	0.8749	*0.0002
CCT	Pro	13.5	1.2711	*0.0001	CAA	Gln	27.3	0.8722	*0.0001
TCA	Ser	18.7	1.2642	*0.0001	GCC	Ala	12.6	0.8607	*0.0001
TAC	Tyr	14.8	1.2515	*0.0001	GAC	Asp	20.2	0.8506	*0.0001
TAT	Tyr	18.8	1.2506	*0.0001	TGT	Cys	8.1	0.8126	*0.0001
GAG	Glu	19.2	1.2465	*0.0001	GCT	Ala	21.2	0.809	*0.0001
CTA	Leu	13.4	1.246	*0.0001	ATC	Ile	17.2	0.8044	*0.0001
CTT	Leu	12.3	1.2375	*0.0001	ACT	Thr	20.3	0.7776	*0.0001
TGC	Cys	4.8	1.2281	*0.0001	GAT	Asp	37.6	0.7569	*0.0001
GGC	Gly	9.8	1.215	*0.0001	AAC	Asn	24.8	0.7564	*0.0001
CAG	Gln	12.1	1.1505	*0.0002	GTT	Val	22.1	0.7544	*0.0001
ACG	Thr	8	1.1164	0.0069	GTC	Val	11.8	0.7541	*0.0001
AGT	Ser	14.2	1.1037	0.006	AAG	Lys	30.8	0.7409	*0.0001
AGC	Ser	9.8	1.0916	0.0213	ACC	Thr	12.7	0.6969	*0.0001
CAC	His	7.8	1.0815	0.0098					

Table 5.3: Ribosome residence time at position 5. Usage of each codon per 1000 codons and the ribosome residence time (RRT) at position 5 (the P-site of the ribosome). The p-value for a difference between the calculated RRT value and an RRT value of 1 is shown. P-values less than or equal to 0.001 are marked with an asterisk.)

Codon	AA	Usage	RRT	p-value	Codon	AA	Usage	RRT	p-value
CCT	Pro	13.5	1.8011	*0.0001	GGG	Gly	6	0.9566	0.1321
CCC	Pro	6.8	1.47933	*0.0001	ACG	Thr	8	0.9507	0.1391
CCA	Pro	18.3	1.4771	*0.0001	CGT	Arg	6.4	0.946	0.0431
AAT	Asn	35.7	1.3892	*0.0001	TTC	Phe	18.4	0.9401	0.0064
CGG	Arg	1.7	1.3446	0.007	TGC	Cys	4.8	0.9379	0.1141
CCG	Pro	5.3	1.302	*0.0001	AAG	Lys	30.8	0.9366	0.0062
CAT	His	13.6	1.29	*0.0001	GCC	Ala	12.6	0.9347	0.0025
GGT	Gly	23.9	1.1917	*0.0001	GAA	Glu	45.6	0.917	*0.0001
AAC	Asn	24.8	1.18	*0.0001	ACC	Thr	12.7	0.9144	*0.0009
GAT	Asp	37.6	1.1767	*0.0001	ATC	Ile	17.2	0.9071	*0.0002
CGA	Arg	3	1.146	0.1353	AGA	Arg	21.3	0.8999	*0.0002
GTA	Val	11.8	1.1364	*0.0006	GTT	Val	22.1	0.8667	*0.0001
GGA	Gly	10.9	1.0954	0.0083	CTT	Leu	12.3	0.8664	*0.0004
GGC	Gly	9.8	1.0913	*0.0009	CTG	Leu	10.5	0.8471	*0.0001
ACT	Thr	20.3	1.0877	*0.0004	TAC	Tyr	14.8	0.8436	*0.0001
TTA	Leu	26.2	1.0751	0.0083	GTC	Val	11.8	0.8131	*0.0001
AAA	Lys	41.9	1.0594	0.03	ATG	Met	20.9	0.812	*0.0001
CAA	Gln	27.3	1.0578	0.0137	TTG	Leu	27.2	0.8093	*0.0001
ATT	Ile	30.1	1.057	0.0106	AGT	Ser	14.2	0.8011	*0.0001
TGT	Cys	8.1	1.0494	0.0701	ATA	Ile	17.8	0.7969	*0.0002
TTT	Phe	26.1	1.0459	0.059	TCC	Ser	14.2	0.788	*0.0001
ACA	Thr	17.8	1.0436	0.1152	TCG	Ser	8.6	0.7871	*0.0001
TAT	Tyr	18.8	1.0424	0.1107	GAG	Glu	19.2	0.7798	*0.0001
GCA	Ala	16.2	1.0286	0.1647	GCG	Ala	6.2	0.7749	*0.0001
GCT	Ala	21.2	1.0252	0.1298	CGC	Arg	2.6	0.7573	*0.0003
TCA	Ser	18.7	1.0213	0.264	GTG	Val	10.8	0.7532	*0.0001
TCT	Ser	23.5	1.0144	0.273	TGG	Trp	10.4	0.7195	*0.0001
GAC	Asp	20.2	0.9988	0.4721	AGC	Ser	9.8	0.7066	*0.0001
CAG	Gln	12.1	0.9986	0.4946	AGG	Arg	9.2	0.7061	*0.0001
CAC	His	7.8	0.975	0.2335	CTC	Leu	5.4	0.6666	*0.0001
CTA	Leu	13.4	0.9643	0.1462					

Table 5.4: Pairwise *Spearman* correlation between the RRT values at position 6 for 4 different data-sets

	$YPD1$	$-His$	$YPD2$	$Ingo.$
-Lys	0.80	0.35	0.76	0.22
YPD1		0.53	0.96	0.55
-His			0.58	0.37
YPD2				0.53

Table 5.5: Top 10 RRTs at positions 3 through 6 of the anisomycin-generated short footprints

AA_{Pos3}	$CODON_{Pos3}$	RRT_{Pos3}	AA_{Pos4}	$CODON_{Pos4}$	RRT_{Pos4}
Gly	GGG	2.64	Pro	CCC	2.36
Gly	GGC	2.52	Pro	CCA	2.34
Gly	GGT	2.36	Met	ATG	2.25
Gly	GGA	2.32	Pro	CCT	2.17
Asp	GAC	1.8	Ala	GCC	2.13
Ala	GCC	1.79	Phe	TTC	2.03
Ala	GCA	1.7	Ala	GCA	2.01
Ala	GCT	1.65	Ala	GCT	1.98
Ala	GCG	1.59	Tyr	TAC	1.98
Glu	GAG	1.58	Ser	TCC	1.97
AA_{Pos5}	$CODON_{Pos5}$	RRT_{Pos5}	AA_{Pos6}	$CODON_{Pos6}$	RRT_{Pos6}
Leu	TTA	2.75	Arg	CGA	3.72
Leu	CTC	2.73	Arg	CGG	3.5
Val	GTA	2.43	Pro	CCG	2.74
Leu	CTA	2.36	Lys	AAA	2.59
Leu	TTG	2.29	Lys	AAG	2.49
Val	GTG	2.21	Arg	CGC	2.46
Leu	CTT	2.16	Arg	CGT	2.34
Val	GTC	2.12	Arg	AGG	2.32
Val	GTT	2.11	Arg	AGA	2.21
Ile	ATA	2.03	Asp	GAT	2.12

codon usage and a high rate of decoding. Although this is a correlation that has been widely expected, there has been little evidence for it; indeed, the most recent experiments suggested that all codons were decoded at the same rate [57, 98]. Some workers have had other expectations for decoding rates. For instance, an important theory was that the more common codons were common because their translation might be more accurate [8] (and this still might be correct).

Translation is optimized for both speed and accuracy [124]. During translation, the ribosome must sample many incorrect tRNAs at the A-site before finding a correct tRNA. It must match the anticodon of that correct tRNA with the codon; after such matching, there is a conformational change around the codon-anticodon interaction at the decoding center [125, 126]. The ribosome must form the peptide bond [127, 128], translocate [119, 120, 129], and eject the empty tRNA. The nascent peptide must make its way through the ribosome exit tunnel [121, 130, 131]. Depending on the rate of each of these events, the concentration of the various tRNAs might or might not have a detectable effect on the overall rate of translation. Our findings that (i) the more frequent codons (i.e., the ones with the highest tRNA concentrations) are decoded rapidly; and (ii) GC-rich codons are decoded slowly; and (iii) *proline* is slow in the P-site, suggest that there are at least three processes that happen somewhat slowly and on a similar timescale. The high rate of decoding for high concentration tRNAs may reflect the relatively short time it takes for the ribosome to find a high-concentration correct tRNA among many incorrect tRNAs. The fact that we detect *proline-specific* delays of a similar magnitude to the rare-codon specific delays suggest that peptide bond formation and identification of the correct tRNA are happening on similar time scales. In general, this is what one might expect from the evolution of such an important process as protein synthesis - if one process were entirely rate-limiting, there would be very strong selection for greater speed in that process, until a point is reached where it catches up with other processes, and several processes together are then rate-limiting.

Even though these data establish that common codons are translated relatively rapidly, this does not on its own explain the success of codon optimization for increasing protein expression, since the rate of translation is primarily limited by the rate of initiation, not elongation [8, 132] (although one recent study identifies a mechanism whereby rapid elongation causes rapid initiation [133]). Nevertheless, on a genome-wide (and not gene-specific) scale, the use of faster codons would mean that a given genomic set of mRNAs would require (or titrate out) fewer ribosomes to make a given amount of protein than the same set of mRNAs using slower codons [8, 132]. Based on our RRT measure-

ments, and taking into account the different copy numbers of different mRNAs [60], we roughly estimate that yeast require about 5% fewer ribosomes than if they were to make protein at the same overall rate but using each synonymous codon at an equal frequency (see sec. 5.5). This provides at least a sufficient reason for the bias towards faster synonymous codons.

We applied RRT analysis to the short footprints identified by Lareau et al. (fig. 5.7). These short footprints seem to report on a different translational process than the long footprints seen in cycloheximide experiments. We see that the basic amino acids *Arg* and *Lys* are slow at position 6; small hydrophobic amino acids are slow at position 5; and glycine is slow at position 3. While we know too little about the nature of the short footprints to reliably interpret these results, one speculative possibility is that the results report on the interaction of amino acids in the nascent peptide chain with the exit tunnel of the ribosome [130, 131, 134, 135]. We find *Arg* and *Lys* slow at position 6, and this correlates with the fact that these basic amino acids cause a pause by interacting with the exit tunnel [98, 99, 121]. This would then suggest that small hydrophobic amino acids, and then glycine, might similarly cause pauses by interacting with positions one or three amino acids further out in the exit tunnel.

In summary, we believe that RRT analysis is a sensitive, high-resolution method that can characterize the interaction of codons and amino acids with the ribosome. It can be applied to ribosome profiling data of many types, from many organisms. Here, we show that frequent codons are decoded more quickly than rare codons; that codons high in AT are decoded somewhat quickly; that *proline* forms peptide bonds slowly; and that short footprints from anisomycin treated cells have an interesting RRT profile that may reflect interaction of amino acids with the ribosome exit tunnel.

5.5 Materials and methods

Experiments were done with yeast strain background BY4741. Ribosome profiling was based on the method of Ingolia [85], but with modifications (see below). Programs for analysis of ribosome residence time were written by the authors (primarily R.Y. and A.Y.). The code for ribosome residence time analysis is available from the authors upon request.

5.5.1 Ribosome profiling

Informatic analysis was conducted on four ribosome profiling experiments (YPD1, YPD2, SC-lys, and SC-his) done for other reasons in the Futcher lab.

The strains and methods used varied slightly from experiment to experiment; nevertheless similar results were obtained for the RRT analysis table 5.5. The ribosome profiling experiments YPD1 and YPD2 have been reported previously (Cai and Futcher, 2013) as the *WT* and *whi3* experiments, respectively.

All experiments used *S. cerevisiae* strain background BY4741. Two biologically independent ribosome-profiling libraries and mRNA-seq libraries were obtained from YPD rich media (the YPD1 and YPD2 experiments), and two biologically independent ribosome-profiling libraries and mRNA-seq libraries were prepared in synthetic complete media (the SC-lys and SC-his experiments). Two methods for harvesting cells were used. After harvesting and footprint size selection, footprints from all four experiments were processed identically into sequencing libraries using the ARTseq Yeast Ribosome Profiling kit, following the manufacture's instructions beginning with step B3 in the protocol.

- *Harvesting method 1 (YPD1 and YPD2 experiments):* – 1 liter of cells in YPD were grown to a density of $2.0 \times 107 cells/ml$. Medium was cooled to $0°C$ by adding ice (stored at $-20°C$) and simultaneously cycloheximide was added to a concentration of $100\mu g/ml$ to quickly halt translation and freeze translating ribosomes in place. Cells were centrifuged using a Sorvall Evolution RC centrifuge at $3000rpm$ for 2 minutes at $4°C$. The resulting cell pellet was washed with ice-cold RNase-free water containing $100\mu g/ml$ cycloheximide by gentle vortexing, and repelleted. Supernatant was aspirated, and cells were re-suspended in polysome lysis buffer prepared according to the ARTseq ribosome profiling kit instructions. Cell lysis buffer slurry was slowly dripped into an RNase-free $50ml$ conical tube containing liquid nitrogen. Resulting frozen pellets of cell slurry were lysed using a TissueLyser II and $50ml$ grinding jars at liquid nitrogen temperature for six 3 minute cycles at 15 hertz. Frozen cell lysate was scraped from the grinding jar into a new RNase-free $50ml$ conical tube followed by reheating the slurry in a $30°C$ waterbath with constant swirling. Immediately after complete thawing ($\sim 3-5$ minutes), cell lysate was centrifuged for 5 minutes at $3000 \times g$. Supernatant was moved to a $1.5ml$ RNase-free centrifuge tube and centrifuged for 10 minutes at $20,000 \times g$. Clarified lysate total RNA content was estimated using a Nanodrop at A260nm, and polysome complexes were digested using ARTseq ribonuclease mix according to the manufacture's instructions. Ribosome-protected mRNA footprints were purified using an Illustra Microspin S-400HR column prepared according to ARTseq manufacture's instructions. All following library generation steps were performed according to the ARTseq protocol starting at step 4 (PAGE

purification). Following the end repair step in the protocol, a biotinylated oligonucleotide antisense to a specific rRNA fragment was used to reduce rRNA contamination using a protocol from the Jonathan Weissman lab (personal communication from Gloria Brar).

- *Harvesting method 2 (SC-lys and SC-his experiments):* – Synthetic media lacking lysine or lacking histidine was used to prepare 1 liter of cells at $2.0 \times 107 cells/ml$. The strains were prototropic for *Lys* or *His* (*HIS3* gap1 frame1), respectively. Cells were harvested by vacuum filtration using Whatman 7184-009 membrane filters at $30°C$. A liquid nitrogen cooled spatula was used to scrap cells from the membrane followed by immediate flash freezing in an RNase-free $50ml$ conical tube containing liquid nitrogen. Special care was taken to ensure cells were exposed to air for as little time as possible between vacuum filtration and flash freezing ($2-3$ seconds) to prevent loss of ribosome footprints at the 5' ends of mRNAs (personal communication, Gloria Brar). ARTseq polysome lysis buffer containing cycloheximide at $50\mu g/ml$ was slowly dripped into the liquid nitrogen filled cell pellet conical tube. Cells were lysed using a TissueLyser II and 50ml grinding jars at liquid nitrogen temperature for six 3 minute cycles at 15 hertz. Frozen cell lysate was scraped from the grinding jar into a new RNase-free $50ml$ conical tube followed by reheating the slurry in a $30°C$ waterbath with constant swirling. Immediately after complete thawing ($\sim 3-5$ minutes), cell lysate was centrifuged for 5 minutes at $3000 \times g$. Supernatant was moved to a $1.5ml$ RNase-free centrifuge tube and centrifuged for 10 minutes at $20,000 \times g$. Clarified lysate total RNA content was estimated using a Nanodrop at A260nm, and polysome complexes were digested using ARTseq ribonuclease mix according to the manufacture's instructions.

- *SC-lys dataset:* – Digested monosomes were purified using sucrose cushion ultracentrifugation for 3 hours at $35,000rpm$ using a SW-41 rotor. The sucrose cushion contained $9ml$ of 10% sucrose polysome lysis buffer lacking triton detergent layered over $3ml$ of 60% sucrose polysome lysis buffer lacking triton detergent. Gradient fractionation was carried out using a BioRad EM-1 UV absorbance monitor and peristaltic pump. Efficiency of RNase digestion was monitored in tandem using a undigested control lysate on an identically prepared 10%-60% sucrose cushion and a digested control centrifuged on a 10%-60% sucrose gradient. Following fractionation, the monosome containing fraction was mixed 1:1 with 4M guanidine thiocyanate and was precipitated overnight using a 1:1 volume of 100% isopropanol chilled to $-20°C$. The RNA pellet was aspirated

and resuspended in 400μl RNase-free water and protein was removed by two acid phenol-chloroform purifications followed by one chloroform purification. Recovered supernatant was brought to 0.3M ammonium acetate and precipitated with 3 volumes of 100% ethanol. All following library generation steps were performed according to the ARTseq protocol starting at step 4 (PAGE purification). Following the end repair step in the protocol, a biotinylated oligonucleotide antisense to a specific rRNA fragment was used to reduce rRNA contamination using a protocol from the Jonathan Weissman lab (personal communication Gloria Brar).

- *SC-his dataset:* – Digested monosomes were purified using an Illustra Microspin S-400HR column according to ARTseq manufacture's instruction. All following library generation steps were performed according to the ARTseq protocol starting at step 4 (PAGE purification). Following the end repair step in the protocol, a biotinylated oligonucleotide antisense to a specific rRNA fragment was used to reduce rRNA contamination using a protocol from the Jonathan Weissman lab (personal communication Gloria Brar).

5.5.2 Data analysis

Unless indicated, data processing and analysis was performed using a collection of custom programs written in *Perl*.

- *Sequence processing and alignment:* – Primary data was generated using Illumina HiSeq2000. Data was processed using Fastq clipper from the FASTX Toolkit 0.0.13 to remove the adaptor sequence and all reads shorter than 25 nucleotides were discarded. Alignment to the reference was done using bowtie2 2.1.0 in local alignment mode.

  Before performing our analysis on the Ingolia et al. data [85], in order to adhere to the processing guidelines of that paper, we used bowtie 0.12.8, reporting all alignments with at most 3 mismatches, and a seed length of 21. We then processed the multiple alignments, removing the poly-A tails and picking the one with the greatest number of bases matching to the reference.

- *Ribosome residence time analysis:* – This analysis uses the general idea that many different mRNA sequences should get an independent and equal vote on decoding speed. We opted to analyze select regions where the effects of codon usage become particularly easy to assay. First we

discounted all reads with more than 2 mismatches or quality less than 10. We identified the first in-frame codon of each read, and discarded those less than 30 nucleotides long to exclude fragments that may have been over digested by RNAase I. We then examined the coding regions of the genome, ignoring those overlapping with other genes, rRNAs, and tRNAs, in order to maximize our confidence in unique mapping. Each of the footprint reads that fully fit into a coding region that it aligned to was considered for further analysis.

For each particular codon, we identified all instances in our coding regions where this codon (say CTC) occurs uniquely within a window of ten codons upstream and ten codons downstream (i.e. a window of 19 codons with the target CTC in the center of the window). For footprints that are 10 codons long, there will be ten classes of footprints where this particular CTC can appear - position 1, position 2, . . . position 10. Thus, all footprints where the first codon of the footprint aligns to this particular CTC will belong to the position 1 class, all footprints where the second codon of the footprint aligns to this particular CTC will belong to position 2 class, etc.

In the absence of any codon preference of the ribosome, we would expect to see a uniform distribution of reads across these ten classes. In general, the codon-positional preference is described by the relative frequency of reads in each of these classes. These relative frequency distributions can be fairly averaged over all target regions over all genes centered on a specific codon. This average we call the "Ribosome Residence Time" (RRT); it is intended as a statistical estimate of the relative time spent by the ribosome at a particular codon at a particular read position. Typically we discuss the RRT at position 6 (the A-site), but we also discuss the RRT at position 5 (the P-site). Regions on highly expressed genes can be fairly compared with similar regions on genes with lower expression, because we are dealing with relative frequency distributions (i.e. percentage instead of read counts). Each region represents an independent trial of any positional preference of the given central codon. Averaging over the hundreds or thousands of occurrences on the genome provides for a statistically rigorous analysis.

Relative frequency distributions will only be representative if the observed number of reads in the window is high enough that no single position dominates the distribution. For this reason, we restricted our analysis to windows with at least 20 total reads with at least 3 non-empty classes.

The frequency distributions are not normally distributed; this is in part because the number of reads is limited, so many windows have zero footprints at many positions, so the mode of the distribution is often 0. Nevertheless we believe that the mean is a good summary statistic. Maximum values are less than 1, so the mean cannot be skewed by extremely high values. We have also calculated the RRT using the median of the windows instead of the mean, but the results are almost indistinguishable. The *Spearman* rank correlation between the RRT as calculated by the mean, and by the median, is 0.97, while the *Kendall Tau* correlation is 0.89.

For each codon, we obtain the two-tailed p-value by comparing the experimental relative frequency to the distribution of 10,000 relative frequencies based on permuted results. For each of the 10,000 instances, for each considered window, we permute the footprint counts of the 10 position classes.

We performed our RRT analysis on the Ingolia et al. data [85], with small modifications. We did not perform the checks of read quality and the number of mismatches, as this was taken care of in pre-processing steps (See *'Sequence processing and alignment'*). We also considered all reads with at least 24 nucleotides, and performed our relative frequency calculations on the 8 codons, because the majority of the reads were shorter than the reported size selection of RNA fragments $\sim 27 - 31$ nucleotides in length.

The statistical significance shown in table 5.2 and table 5.3 were obtained by constructing 10,000 random simulated frequency distributions and independently permuting each region's frequency distribution prior to averaging. The rank of each observed positional peak among these simulated distributions established the p-value.

- *Codon coherence analysis:* – We developed a p-value computation to assess whether the codons for a given amino acid behave similarly to one another (i.e., are coherent) or not. Each codon's RRT values along the positions of a footprint may be considered as a k-dimensional vector, where k is the number of positions in the footprint (10 for long reads vs. 7 for short reads). Each point in the vector represents a position in the k-dimensional space. Coherency analysis was performed on the set of points at each dimension belonging to the synonymous codons of a particular amino acid. For any given set of c such points, we can compute the average pairwise distance d between them over all $c \times (c-1) \div 2$ pairs of points. If all codons for an amino acid behave similarly, then the points

are close together, and the distance d is relatively small, indicating codon coherence (amino-acid specific behavior), whereas if the various codons for a given amino acid behave differently (non-coherent, codon-specific behavior), then the distance d is relatively large.

To judge the sizes of these distances for a particular set of points, S, containing c codons (c ranges from 2 to 6) for a particular amino acid, we used a p-value. We constructed 10,000 random samples of c codons drawn from the 61 possible sense codons. For each sample, we computed the average pairwise distance and compared this to the average pairwise distance of S. The rank of S in this distribution provides a p-value, which is significant if the vast bulk of random samples have greater pairwise distance than S. Results are shown in fig. 5.8.

- *Estimates of ribosomes needed for differently-encoded transcriptomes:* – An mRNA encoding a given protein could use only the fastest codon for each amino acid, or only the slowest, or it could use a mixture. In each case, the mRNA would occupy, or titrate out, a different number of ribosomes. A transcriptome of mRNAs using only the slowest codons would require more ribosomes to make a given amount of total protein in a given time than a transcriptome of mRNAs using only the fastest codons. We roughly estimated the size of this effect for the range of codon decoding speeds we observed. We generated in silico a yeast transcriptome using only the fastest codon for each amino acid at position 6 (from table 5.1), or only the slowest codon, or a random mixture of codons. Furthermore, we weighted the abundance of each mRNA according to its actual abundance as measured by Lipson et al. [60]. We then compared the relative time required to translate each of these in silico transcriptomes by a set number of ribosomes based on the RRT values for each codon at position 5 and 6, and also assuming that the relevant delay is the delay at position 5 plus the delay at position 6 (since these two reactions must occur sequentially and not simultaneously before the ribosome can shift along the mRNA). In doing this, we noted that the RRT values for position 5 are negatively correlated with those at position 6. Results are as follows: the random encoding requires 1.05 as long as WT; the slowest encoding requires 1.168 as long as WT; and the fastest encoding requires 0.930 as long as WT. Note that this estimate uses the simplification that each species of mRNA will initiate translation at the same rate. A more accurate calculation in which the more abundant mRNAs initiate more rapidly than average would increase the difference between the WT and the random encodings.

5.6 Author contributions

Experiments were conceived and designed by JG, SS, RY, AY and BF. Wet lab experiments were done by JG and YC. Algorithmic design was by SS. RY and AY implemented algorithmic design as code, with some contribution from JG. All authors contributed to interpretation of data. The paper was written largely by BF, AY and JG with contributions from all other authors.

5.7 Acknowledgments

We thank J. Weissman and G. Brar for their generosity in helping us learn ribosome profiling, and for providing protocols and advice. Three anonymous reviewers provided insightful comments that greatly improved the final manuscript. This work was supported by NIH grant R01 GM098400 to BF, and NSF grants DBI-1060572 and IIS-1017181 to SS.

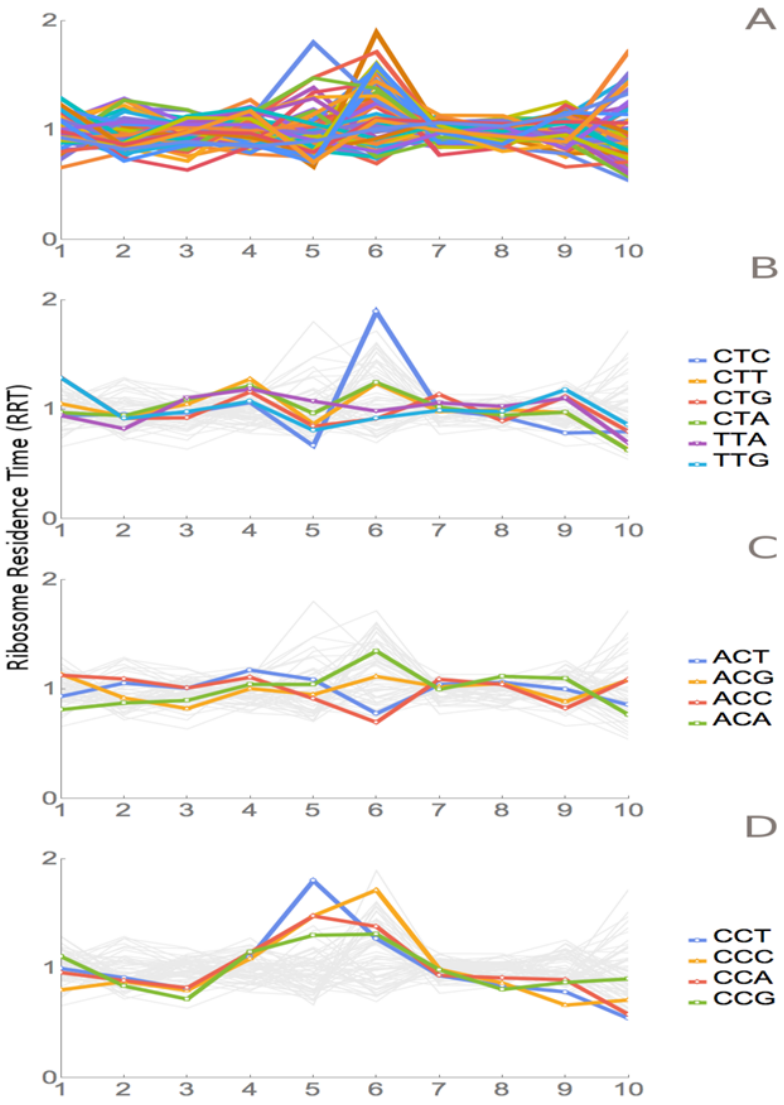

Figure 5.4: Ribosome residence times. (A) The pattern of RRTs for all codons at all positions of the 30 nt footprint reads. Most peaks are at position 6, with some at position 5. (B) The RRTs for the six leucine codons. Among all codons, *CTC* has the highest RRT at position 6. (C) The RRTs for the four threonine codons. *ACC* has the lowest RRT of all codons at position 6. (D) The RRTs for the four proline codons. Proline has peaks at position 5 (the P-site), as well as at position 6.

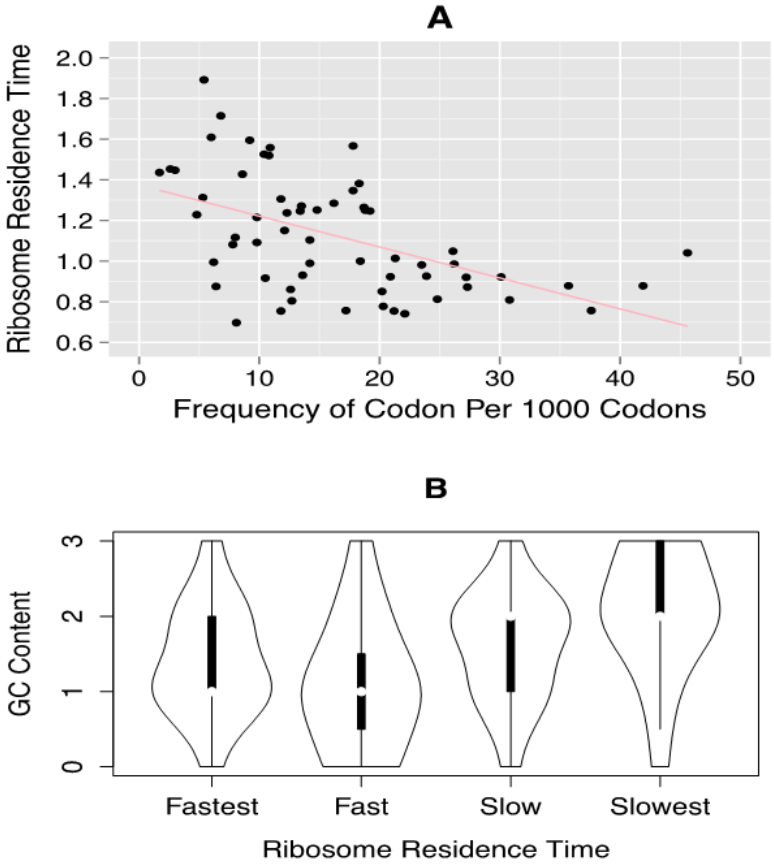

Figure 5.5: Correlation of ribosome residence times with codon properties. (A) Correlation of RRT with codon usage. RRT is plotted against the frequency of each codon per 1000 codons. (B) Correlation of RRT with the GC content of each codon. The codons were divided into quartiles by RRT, $Fastest - lowest$ (around 15 codons per group), and the GC content of the codons in each group is shown in a violin plot.

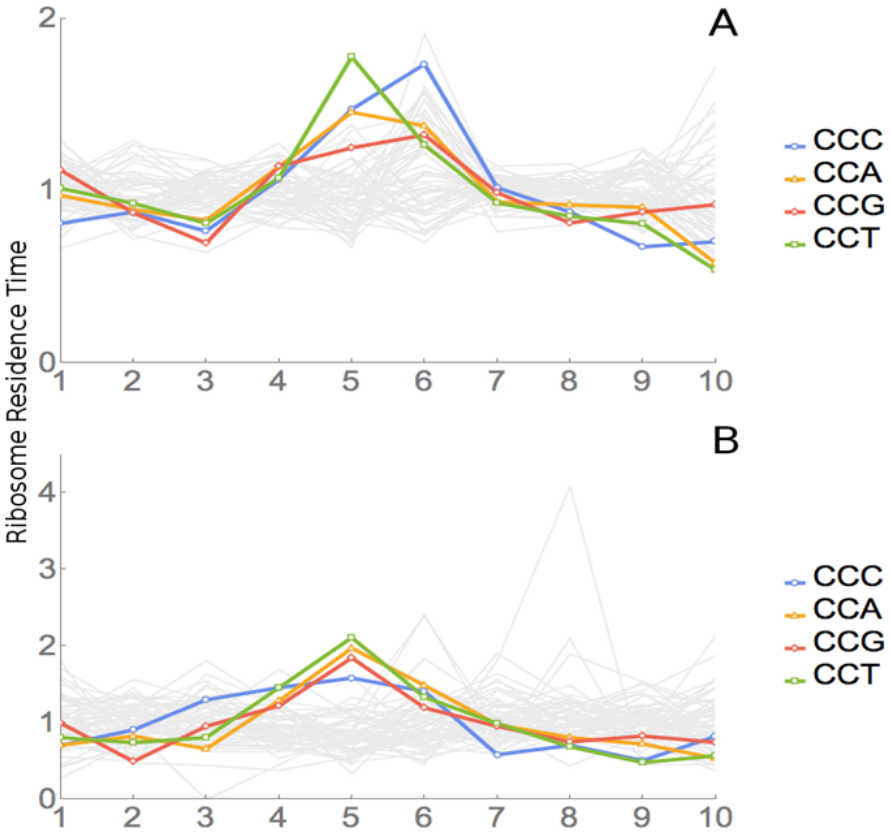

Figure 5.6: Analysis of *ProPro* dipeptides. (A) RRT analysis of the windows containing no ProPro dipeptides. (B) RRT analysis of the windows containing ProPro dipeptides.

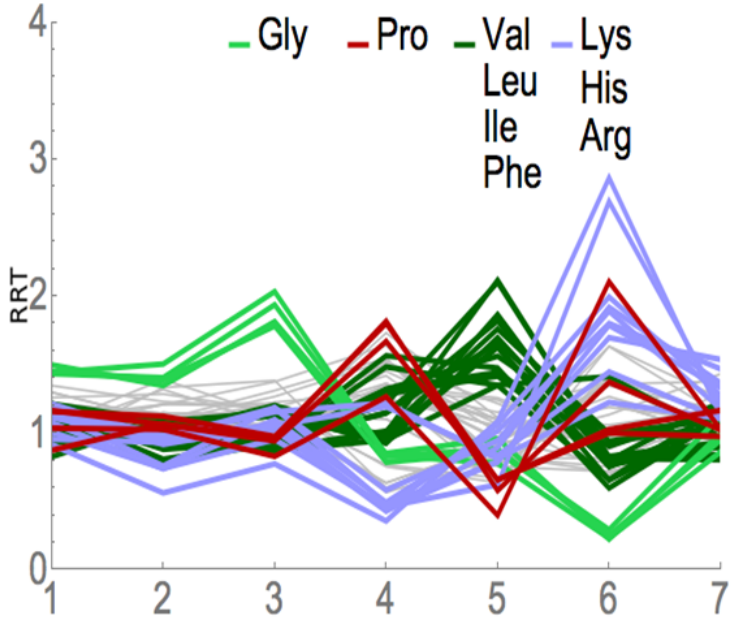

Figure 5.7: RRT analysis of short footprints from anisomycin treatment. The short, seven-codon footprints from anisomycin treatment from Lareau et al. [87] were analyzed for RRT. All 61 sense codons are shown; codons from selected amino acids are color-coded. Position along the footprint is shown on the x-axis.

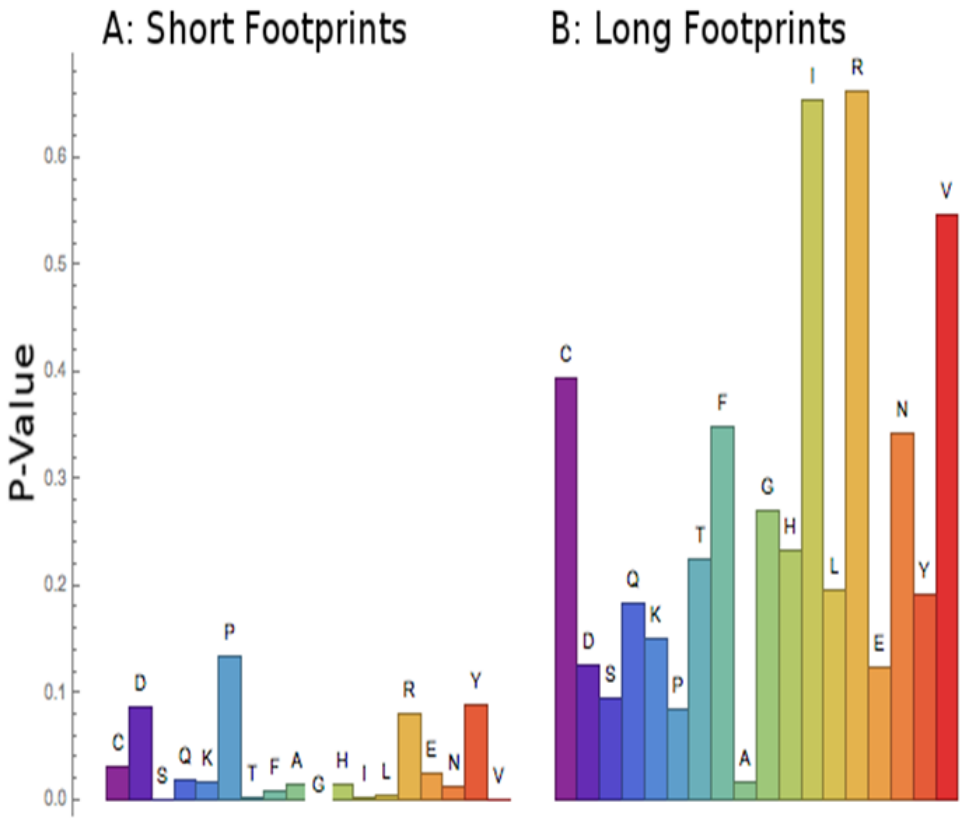

Figure 5.8: Short footprints are amino-acid specific; Long footprints are codon specific. For the set of codons corresponding to each amino acid (x-axis), a test was done to see if all codons behaved similarly or not. For the short footprints (left), p-values (y-axis) are generally small, showing that codons belonging to the same amino acid behave similarly. For the long footprints (right), p-values are generally large, which shows that the synonymous codons (corresponding to same amino acid) behave differently.

Chapter 6

Statistical analysis of ribosome profile data[1]

Ribosome profiling is an emerging technique for identifying active ribosome positions bound to the target mRNA. Here, fragments of a translating mRNA (that are covered by ribosomes) are isolated using the method of nuclease digestion - which degrades unprotected mRNA regions. A translating ribosome surrounds around 30nt positions of an mRNA (may vary based on the conformation of the ribosome, see chapter 5 for detail, [87]). By analyzing the protected area of the translating mRNA, exact ribosome position and the translated message can be detected.

Ribosome footprint data can be used for various applications including the discovery of novel ORFs, making revisions to the gene annotations, genomic expression measurement and in studying the mechanics of translation and of co-translational processes in vivo [100]. The rate of translation varies across the length of an mRNA. The ribosomal pauses in the translation process can regulate synthesis, folding and localization of a protein or mRNA [97, 136, 137, 138, 139, 140, 141]. Several factors could contribute to these pauses, such as codon usage, RNA folded structure or the peptide sequence [142, 143, 144]. tRNA re-use can also be a contributing factor in boosting the translation efficiency [10].

We work on the high coverage ribosome profile data to answer these questions. In this chapter, I will describe our work on predicting impact of the bias introduced by the codon-pair preferences, on correlating ribosome profile data with the folded mRNA secondary structure around the ribosome and finally on determining the impact of tRNA auto-correlation on the ribosome residency time. Ribosome profile data generation steps are explained in detail

[1]Results not published yet.

in sec. 5.5. Profile data were analyzed using python. For data pre-processing and sam file generation FASTX Toolkit 0.0.13 and bowtie2 were used, and python package HTSeq [145] was used for mapping reads to genes.

6.1 Codon-pair bias analysis

In our work, we have generated high-coverage ribosome profile data in *yeast*, and novel algorithm procedures were employed to analyze the data. We worked on measuring average decoding rate of each individual codon and showed the average decoding rates of the codons are well-correlated with the usage bias (sec. 5.3). However, this may not be the only contributing factor influencing codon usage preference, the context surrounding a codon may also play a role here [146]. The context of a codon may impact translation efficiency and accuracy [142, 146, 147]. Codon context may also play a role in suppressing missense and premature stop codons [148, 149].

Following the study of context-dependent codon bias effect, researchers were interested in studying the effect of codon pair bias on translation efficiency [150, 151]. Gutman et al. found extreme codon pair specific utilization bias in bacteria, yeast, and mammals [150]. Based on the study of 237 *E.coli* protein coding genes, they showed that certain codon pairs were over-represented compared to the theoretically predicted means, while some others were under-represented. Following this study Irwin et al. found that, over-represented codon pairs in protein coding sequences (pairs appearing more frequently than expected) are translated slower than the corresponding under-represented codon pairs [151]. During the time of peptide bond formation, simultaneous accommodation of two tRNAs at the ribosomal A and P sites are required. Hence, compatibilities of adjacent tRNAs at these two sites should play a role in translation rate determination and it may depend to some extent on the nature of the bond formation [152, 153, 154]. Buchan et al. found that, the tRNA used at the ribosomal A site has a strong influence in determining pairing preferences [155].

However, the translation efficiency of a particular codon-pair is independent of the frequency of usage of individual codons in that pair [9, 156]. Gutman et al. demonstrated that, codon pair bias is directional (bias for pair A-B and pair B-A are independent of each other) and restricted to adjacent codons [150, 157]. They also suggested that, highly expressed genes tend to avoid over-represented codon pairs (along with avoiding infrequent codons [158, 159]). The effect of codon-pair bias along with codon usage bias is used in synthetic attenuated virus engineering (*SAVE*) [9]. It would be interesting to study the impact of codon-pair bias on ribosome residency time.

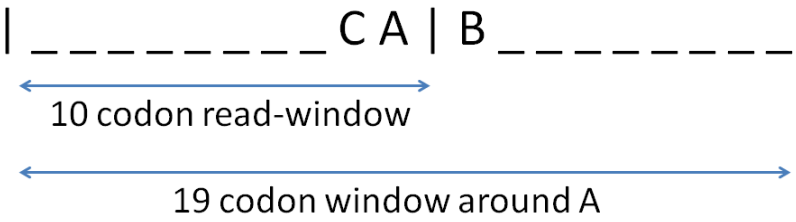

Figure 6.1: 19 codon window with A at the tenth position. We collect reads at the window with A being in the first position, second position, ..., tenth position. Assign the collected data to two different codon-pairs: CA and AB, where CA corresponds to the data-set with reads collected at the second codon of the pair (*set2*) and AB corresponds to the set with reads collected at the first codon of the pair (*set1*).

If we try to analyze the effect of each individual codon-pair on ribosome footprint pileup, our analysis outcome may suffer from the biases introduced by too few footprint reads for some codon-pairs. Moreover, usage bias of individual codons may dominate over the combined effect of a particular codon pair. Hence, we came up with a novel idea to naturally normalize out the impact of codon usage bias effect on the codon-pair bias analysis.

6.1.1 Data collection

Consider a 19-codon window around codon A (fig. 6.1). Suppose the codon ahead of A is C (at position 9 of the window) and the codon following A is B (at position 11 of the window). We collect reads at the window, with A being in the first position (10 codon read-window starting at position 10 of the 19-codon window), second position, ..., tenth position (10 codon read-window starting at the first position of the 19-codon window). Data is collected in a way similar to that explained in sec. 5.5 of chapter 5.

We assign reads at A to two different pairs CA and AB. Reads for CA goes into one data-set (*set2*, which specifies reads at C_2 for all pairs of the form C_1C_2) and reads at AB goes into another data-set (*set1*, which specifies reads at C_1 for all pairs of the form C_1C_2). This way, we collect data for all the 3721 codon pairs (excluding pairs with stop codons) into two sets - *set1* and *set2*.

6.1.2 Data analysis

We considered 3 different codon-pair bias scoring methods:

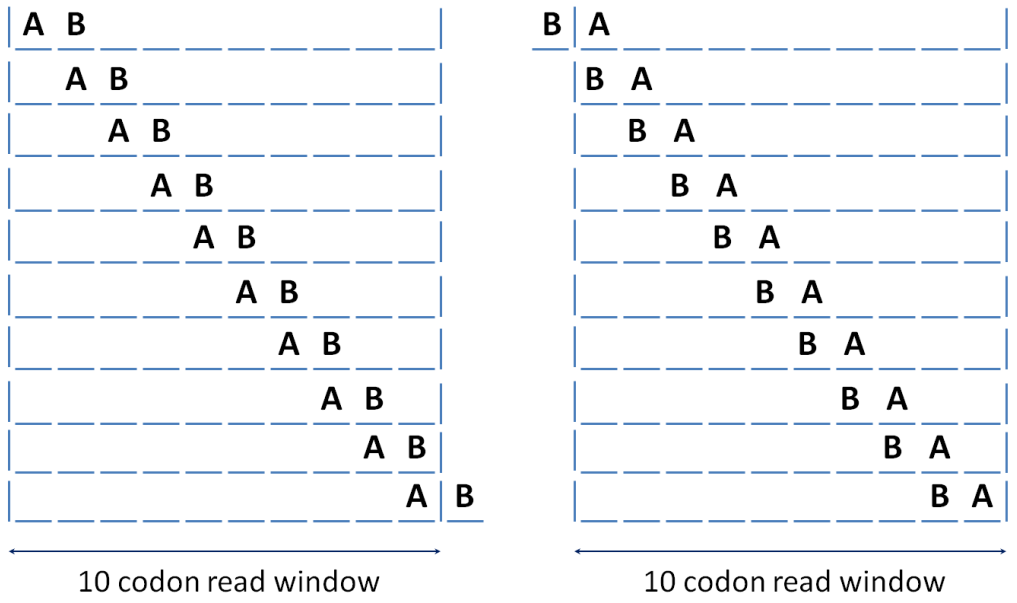

Figure 6.2: Codon pair bias analysis. Consider a read-window of 10 codon length. (A) Codon A is immediately being followed by B. (B) Codon B is immediately ahead of A. We collect data at the window with codon A being in position 1, 2, ..., 10. Collected data is normalized and assigned to *set1* for pair AB and to *set2* for pair BA. Reads at A from *set1* will differ from the reads at A from *set2* based on the bias introduced by the pairs AB vs BA.

- *Classic* - Ranks codon-pairs based on the abundance of the pair in-frame.

  $Score_{Classic} = Score_{Frame0}$

- *Gap* - Assigns scores to codon pairs based on the abundance in-frame vs out of frame.

  $Score_{Gap} = Score_{Frame0} - (Score_{Frame1} + Score_{Frame2})/2$

- *Signal* - Ranks codon-pairs by the average score in all three reading frames.

  $Score_{Signal} = (Score_{Frame0} + Score_{Frame1} + Score_{Frame2})/3$

  The formula to calculate the score of a codon pair at a particular reading frame (frame - 0/ 1/ 2) is as follows [9]:

$$Score_{AB} = \ln \frac{F(AB)}{\frac{F(A)\times F(B)}{F(X)\times F(Y)} \times F(XY)}$$

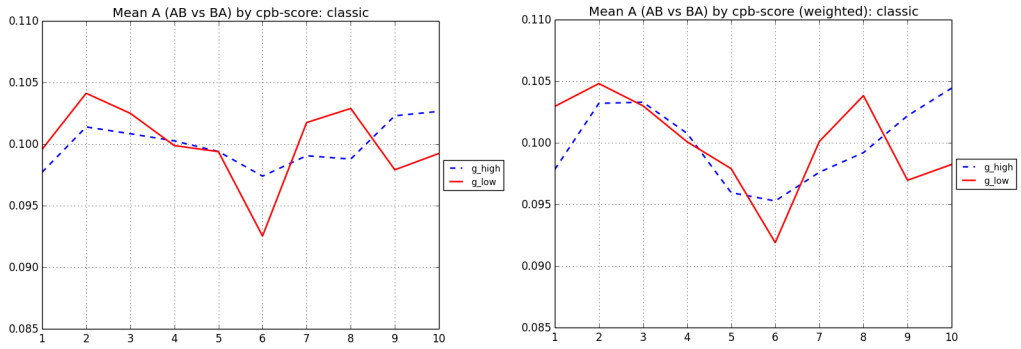

Figure 6.3: Mean codon pair statistics of *high* vs *low* group: *classic* scoring method. The number of codon pairs in each group were 207. Left: Equal weight analysis. Right: Weighted by frequency.

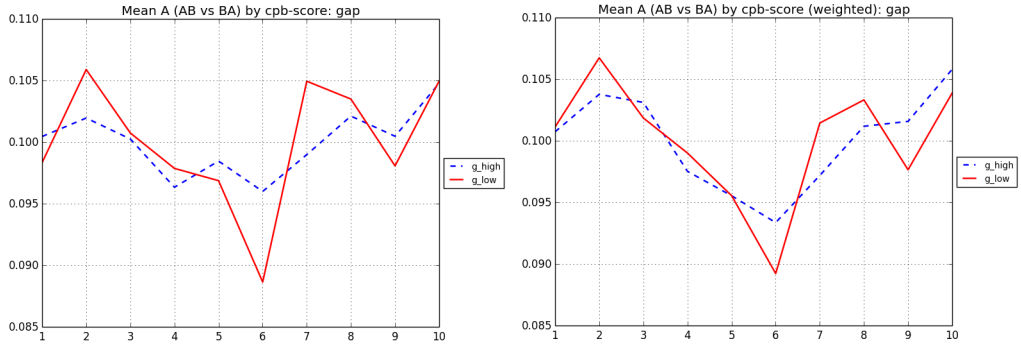

Figure 6.4: Mean codon pair statistics of *high* vs *low* group: *gap* scoring method. 189 codon pairs were considered in each group. Left: Equal weight analysis. Right: Weighted by frequency.

> Here, codons A and B code for amino acids X and Y respectively. F represents the frequency of individual entity being considered at a particular reading frame.

For each of the scoring methods described above, we do the following:

We consider pair of codon pairs of the form AB vs BA, where the scores of these pairs are of opposite sign (if AB is positive, then BA is negative and vice versa). Next, we place these codon pairs into two different groups - *high* vs *low*; the pair with positive score goes into the *high* group and the one with negative score goes into the *low* group. We ignore pairs of the form AA or BB. If the frequency of appearance of any of the codon pairs (AB or BA) is below the minimum threshold frequency (30), we ignore both pairs.

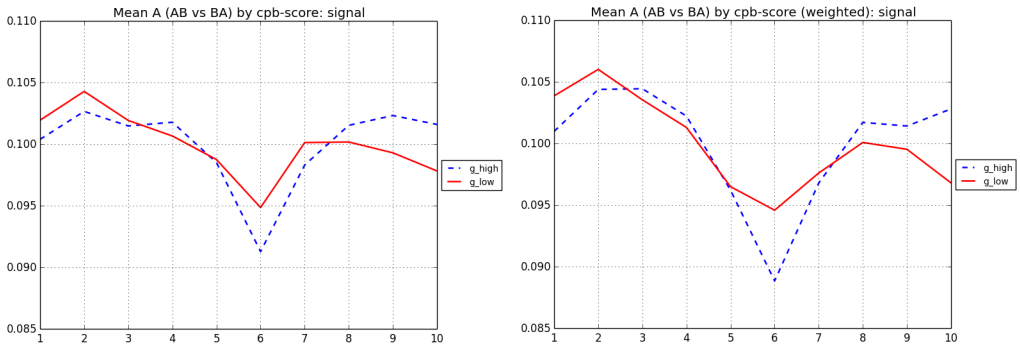

Figure 6.5: Mean codon pair statistics of *high* vs *low* group: *signal* scoring method. Each group had 177 different codon pairs. Left: Equal weight analysis. Right: Weighted by frequency.

Next we calculate mean codon-pair RRT of each group from the previously collected data-sets. For a pair of the form (AB, BA), we consider statistics around A (fig. 6.2). For pair AB we collect data from *set1*, as it specifies reads at C_1 for all pairs of the form C_1C_2. For pair BA, we collect data from *set2*, which specifies reads at C_2 for all pairs of the form C_1C_2. Similarly we collect data for each pair in the two group and then compute the mean score of each group. The way we are doing our analysis, codon usage bias should automatically be nullified or equally distributed in both group, if we assign equal weight to each codon pair. Only difference in the pair-analysis should be based on whether B is ahead or after A. Similar claim can be applied for all other paired codon-pairs (e.g. CD vs DC).

We performed our analysis in two different ways:

- *Equal weight analysis* – Every pair contributes equally. Hence, the mean codon usage RRT scores for both *high* and *low* groups should show similar pattern.
- *Weighted by frequency* – Pairs get weights based on the frequency of appearance. Hence, the patterns for *high* and *low* groups may vary a little based on the difference in the frequency of usage of individual codon pairs.

  For further illustration see sec. B.3.

Now, if codon-pair bias has no impact on the translation process, both *high* and *low* groups should show similar peaks or valleys (similar to codon usage RRT for the groups). However, in our analysis we see different valley depth for two groups in each of the scoring methods. Fig. 6.3 is showing the RRT

statistics of *high* and *low* groups considering *classic* scoring method, which gives positive weight to the in-frame codon pair frequency. 207 different codon pairs were considered in each group. The left plot is showing the statistics for equal weight analysis and the right one is for weighted analysis. Mean RRT at each codon position and the corresponding p-values are listed in table 6.1 and in supplementary table A.9 (p-value computation method is described later). *High* in *classic* represents codon pairs over-represented in-frame, while *low* represents codon pairs under-represented in-frame. Here, we see no significant peak or valley at position 6 of *high* group (p-value = 0.379). However, significantly deep valley is noticed at position 6 for the *low* group (p-value = 0.001). For the weighted analysis, p-values at position 6 of group *high* and *low* are respectively 0.13 and 0.001.

Fig. 6.4 is showing the corresponding statistics considering *gap* scoring method, 189 codon pairs were considered in the analysis. *Gap* assigns positive weights to the in-frame frequency of codon pairs, while giving negative weights to the out of frame frequency of the pairs. Hence, codon pairs in *low* group are under-represented in-frame, while over-represented out of frame. In this case we see similar pattern as the *classic* scoring method, with an even deeper valley at position 6 for the *low* group. For group *high*, p-value at position 6 is 0.784 and for group *low*, it is 0.001 (see table 6.1 for complete statistics). Considering weighted analysis, we get a p-value of 0.284 at position 6 of the *high* group and 0.001 for *low* group (supplementary table A.9).

Fig. 6.5 is showing the RRT statistics considering *signal* scoring method, 177 codon pairs were used in the analysis. For *signal*, *high* group represents codon pairs over-represented in all three coding frames and *low* group has codon pairs under-represented in all three frames. Here, group *high* is showing a significantly deep valley at position 6 (with a p-value of 0.001). However, in case of group *low*, though we see a valley at position 6, it is not significant (p-value = 0.26). Hence, we see a reversal in the pattern for *high* vs *low* group when considering *signal* scoring method. In case of weighted analysis, for the *high* and *low* groups, we get p-values of 0.001 and 0.515 respectively. Complete statistics have been shown in table 6.1 and supplementary table A.9.

Our earlier analysis on measuring the decoding rates of codons showed that, ribosome moves faster over more common codons (chapter 5). For codon pairs, we see the reverse pattern (considering *classic* and *gap* scoring methods). According to Irwin et al., over-represented codon pairs translate slowly compared to under-represented codon pairs [151]. When considering codon pairs under-represented in-frame, our statistics for *low* codon pair group shows similar pattern, much deeper and statistically significant valley compared to the codon pair group *high*. On the other hand, for *signal* scoring method, we see

less reads (or deeper valley) at position 6 of group *high* compared to that for group *low*. As based on the statistics of other two scoring methods, valley depth at abundant codon pair group is insignificant, valley at position 6 for the *signal* scoring method could be because of the codon pairs over-represented out of frame.

However, nature generally optimizes sequences to maximize the amount of protein production. According to Coleman et al., under-represented codon pairs decrease the rate of protein translation [9]. However, faster translation usually leads to increased protein production. One potential reason behind the selection against the under-represented codon pairs could be to control the rate of missense translation. Various in vivo and in vitro experiments suggest that, error in the translation process is very rare and selection acts against the mistranslation [160, 161, 162]. Ribosomal frame-shifting and ribosome bypassing both contribute to the global error frequency of translation process, which are highly controlled by the gene control mechanism. Though frame-shifting can be beneficial for some viruses and retrotransposons [163], these are exceptions to the natural process and occur very rarely (less than 1 per 10000 codons) [164]. Hence, context of the sequences triggering frame-shifting events are highly avoided in the coding regions of the genes. In *Saccharomyces cerevisiae*, hepta-nucleotide sequences triggering +1 frame-shifting are significantly under-represented in the open reading frame [165].

Part of the coding sequences that slow down ribosomal movement (termed as *choke points*) reduces the probability of frame-shifting events [166]. On the other hand, part of coding sequences aiding in faster ribosomal transaction (*slippery sequences*), often cause the translating ribosome to slip and skip few bases, and as a result ends up reading a completely different frame thereafter [Wik]. Hence, valley at position 6 (in fig. 6.3, 6.4, 6.5) can alternately be due to the effect of frame-shifting event. In this latter case, it would mean ribosome slipped more often at those codon pair positions, which resulted in missing reads or lack of reads. Codon pairs with positive scores in *signal* scoring method may have been chosen more often out of frame, to compensate for (or to correct) an accidental shift of the ribosome from the coding frame.

To validate any of the hypotheses explained earlier, further evidence is needed in favor of that hypothesis. It remains as an interesting future challenge to deal with.

6.1.3 P-value computation

Permutation method was used to calculate the p-values at the codon positions of each codon pair group. Rather than assigning codon pairs to *high* vs *low* group based on the abundance of the pairs, pair of codon pairs of the form AB

Table 6.1: Mean RRT of codon pair groups, considering scoring methods: classic, gap, signal.

Scoring method - classic:

Position	high	p-value (high)	low	p-value (low)
1	0.098	0.037	0.100	0.334
2	0.101	0.027	0.104	0.001
3	0.101	0.812	0.103	0.282
4	0.100	0.318	0.100	0.555
5	0.099	0.520	0.099	0.482
6	0.097	0.379	0.093	0.001
7	0.099	0.862	0.102	0.001
8	0.099	0.022	0.103	0.038
9	0.102	0.842	0.098	0.001
10	0.103	0.006	0.099	0.569

Scoring method - gap:

Position	high	p-value (high)	low	p-value (low)
1	0.100	0.474	0.098	0.12
2	0.102	0.470	0.106	0.001
3	0.100	0.904	0.101	0.835
4	0.096	0.349	0.098	0.907
5	0.098	0.518	0.097	0.104
6	0.096	0.784	0.089	0.001
7	0.099	0.941	0.105	0.001
8	0.102	0.829	0.104	0.371
9	0.100	0.818	0.098	0.004
10	0.105	0.218	0.105	0.201

Scoring method - signal:

Position	high	p-value (high)	low	p-value (low)
1	0.100	0.953	0.102	0.753
2	0.103	0.002	0.104	0.001
3	0.101	0.788	0.102	0.642
4	0.102	0.107	0.101	0.438
5	0.099	0.608	0.099	0.705
6	0.091	0.001	0.095	0.26
7	0.098	0.31	0.100	0.146
8	0.102	0.087	0.100	0.475
9	0.102	0.245	0.099	0.06
10	0.102	0.179	0.098	0.064

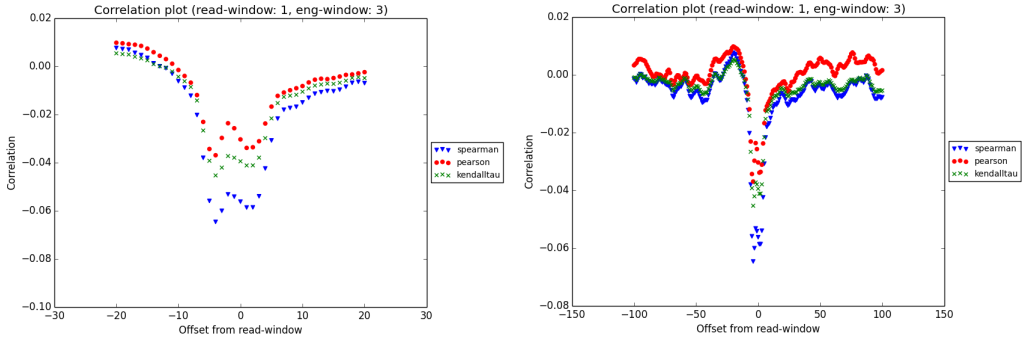

Figure 6.6: *Pearson*, *Spearman* and *Kendall Tau* correlation plot for *Saccharomyces cerevisiae* RNA secondary structure energy vs ribosome footprint reads (read-window size: 1 and RNA secondary structure energy window size: 3). Reads at each gene were normalized by the average reads per codon of that gene. Left: correlation plots for offsets in the range, $-20 <= x <= 20$. Right: correlations for offsets in the range, $-100 <= x <= 100$.

vs BA were assigned randomly to *high* or *low* group.

Next, the mean RRT of *high* group with randomly chosen codon pairs were compared to the mean RRT of the *high* group with abundant codon pairs. Similarly mean RRT of the *low* group with randomly assigned codon pairs were compared to the mean RRT of the *low* group with rare codon pairs. 1000 random permutations of the codon pairs were considered to calculate the p-values of each codon pair group. Table 6.1 shows the mean RRT and p-values of each codon pair group for equal weight analysis. Supplementary table A.9 shows the corresponding statistics considering weighted analysis.

6.2 RNA secondary structure effect on ribosome profile data

Translation of an mRNA is a complex process, which occurs in a non-uniform rate. RNA secondary structure plays critical role in this translation process [8, 12, 13, 167]. Katz et al. [168] studied the biases exhibited by the natural mRNAs to local RNA secondary structure and found that, *Saccharomyces cerevisiae* shows statistically significant biases in favor of local RNA structure measured by the folding energy. Each folded structure in an mRNA sequence must unwind as the ribosome progresses through it. Hence, its speed of traversal may vary based on the strength of the bonded structure. We are interested in predicting the correlation of the RNA secondary structure with ribosome

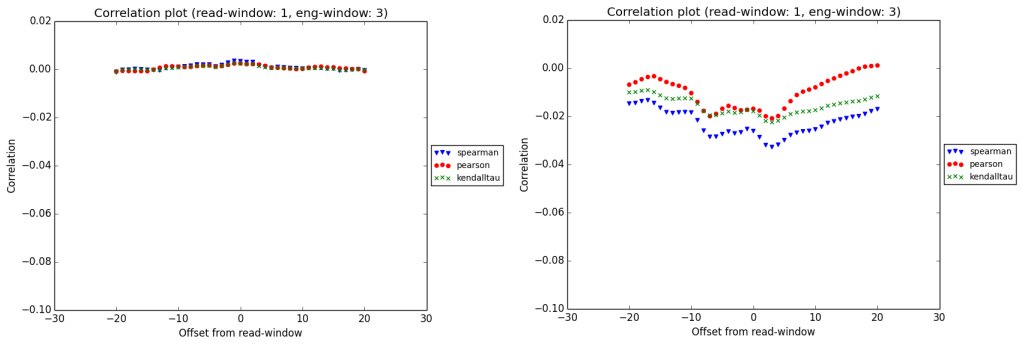

Figure 6.7: Correlation plot for *Saccharomyces cerevisiae* RNA secondary structure energy vs reads per codon at control data-set using *Pearson*, *Spearman* and *Kendall Tau* correlation measures. Left: randomly generated read data-set, no significant pattern in the correlation was observed. Right: wild-type mRNA-seq data-set, the data shows significant two dip pattern similar to the real ribosome profile data-set, but to a lesser extent.

footprint pile-ups.

6.2.1 Data collection and preprocessing

We downloaded PARS assisted folded RNA secondary structures of *Saccharomyces cerevisiae* genes from Segal lab of Computational Biology [169]. Loop based energy of the folded structure is determined using 'RNAeval' from Vienna RNA package. Next we have converted the hierarchical loop based energy obtained from 'RNAeval' to per base energy using our own program. We converted per nucleotide energy to per codon energy by taking the sum of the energies of three nucleotides.

We also have downloaded raw PARS scores, which measures the probability of a nucleotide to be in double-stranded conformation. Higher (more positive) PARS score means increased probability of the nucleotide being in bonded structure, which corresponds to the lower (more negative) energy state of the structure. Hence we considered inverted PARS scores, to make it compatible with the secondary structure energy during the correlation analysis.

We collected raw footprint read counts for each gene and mapped reads to the ribosomal A site (position 6 of a 10 codon-length read, see chapter 5 for detail). For each gene, we normalized reads across the gene by the average reads per codon of that gene.

Table 6.2: Correlation table for *Saccharomyces cerevisiae* RNA secondary structure energy vs ribosome profile data.

Offset	Frequency	Spearman	P-value	Pearson	P-value	K. tau	P-value
-15	226760	0.0034	0.10483	0.0073	0.001	0.0024	0.0885
-14	227260	0.0012	0.57589	0.0058	0.006	0.0008	0.559
-13	227760	-0.0001	0.97756	0.0043	0.040	0.0000	0.972
-12	228260	-0.0009	0.67293	0.0029	0.165	-0.0006	0.652
-11	228760	-0.0032	0.13008	0.0010	0.635	-0.0022	0.11
-10	229260	-0.0063	0.00266	-0.0016	0.438	-0.0044	0.002
-9	229760	-0.0089	1.84E-05	-0.0041	0.05	-0.0063	6.03E-06
-8	230260	-0.0123	3.46E-09	-0.0069	0.001	-0.0087	4.29E-10
-7	230760	-0.0203	1.62E-22	-0.0120	7.41E-09	-0.0143	6.96E-25
-6	231260	-0.0381	5.08E-75	-0.0232	6.88E-29	-0.0268	5.234E-83
-5	231760	-0.0560	2.24E-160	-0.0344	8.97E-62	-0.0393	1.85E-177
-4	232260	-0.0647	5.24E-214	-0.0370	2.67E-71	-0.0454	1.53E-236
-3	232760	-0.0601	2.28E-185	-0.0298	4.95E-47	-0.0422	1.52E-204
-2	233260	-0.0533	2.68E-146	-0.0237	1.98E-30	-0.0374	2.02E-161
-1	233760	-0.0542	8.02E-152	-0.0258	9.06E-36	-0.0380	1.33E-167
0	234260	-0.0563	7.06E-164	-0.0304	3.59E-49	-0.0395	2.93E-181
1	233760	-0.0588	8.66E-178	-0.0340	1.13E-60	-0.0413	7.57E-197
2	233260	-0.0587	4.24E-177	-0.0337	1.26E-59	-0.0412	4.43E-196
3	232760	-0.0541	2.81E-150	-0.0312	2.86E-51	-0.0380	9.97E-167
4	232260	-0.0425	1.88E-93	-0.0239	1.35E-30	-0.0299	8.84E-104
5	231760	-0.0309	5.687E-50	-0.0168	5.35E-16	-0.0217	1.75E-55
6	231260	-0.0218	9.83E-26	-0.0124	2.71E-09	-0.0154	1.53E-28
7	230760	-0.0181	2.96E-18	-0.0110	1.35E-07	-0.0128	3.039E-20
8	230260	-0.0174	6.69E-17	-0.0100	1.49E-06	-0.0123	1.014E-18
9	229760	-0.0169	5.35E-16	-0.0092	1.05E-05	-0.0119	1.08E-17
10	229260	-0.0151	4.95E-13	-0.0083	7.22E-05	-0.0106	2.14E-14
11	228760	-0.0132	2.95E-10	-0.0068	0.0012	-0.0093	2.702E-11
12	228260	-0.0115	4.32E-08	-0.0056	0.007	-0.0081	6.94E-09
13	227760	-0.0108	2.34E-07	-0.0052	0.0123	-0.0076	4.48E-08
14	227260	-0.0103	8.62E-07	-0.0054	0.01	-0.0073	1.88E-07
15	226760	-0.0105	6.27E-07	-0.0050	0.02	-0.0074	1.35E-07

6.2.2 Data analysis

We considered top 500 *Saccharomyces cerevisiae* genes based on the total number of reads per gene. For each gene, we do the following analysis:

Let us consider a read-window of size R and a secondary structure energy window of size S. Now, for each codon-position C of a gene G, we assign the average reads at window $(C - r : C + r)$ to C, where $r = \frac{R}{2}$. Similarly, we calculate average secondary structure energy of the window $(C + x - s : C + x + s)$ for codon C, where $s = \frac{S}{2}$ and x is an offset from the codon position. The idea is to find the strength of the correlation of FTP read with secondary structure energies at different offsets (x) at both $5'$ and $3'$ end of the codon-window being considered.

Three different correlation measures were considered:

- *Pearson correlation* – measures linear correlation (or dependence) between two variables.
- *Spearman's rank correlation* – determines the monotonic relationship between two variables.
- *Kendall's tau correlation* – rank based correlation measure, calculates the strength of the dependence between two variables. For a set of observations of size N: $\tau = \frac{N_c - N_d}{\frac{1}{2}N(N-1)}$, where N_c = number of concordant pairs and N_d = number of discordant pairs.

  Correlation analysis was performed using python *scipy* package *stats*.

The analysis was performed for different values of R and S. Most interesting pattern was obtained for small smoothing windows (i.e. $0 < R <= 3$, $S = 3$). Fig. 6.6 is showing the correlation plot for $R = 1$ and $S = 3$. Here, we have shown correlation values for different offsets (x). The left figure is showing the correlation scores for offsets in the range $-20 <= x <= 20$ and the right one is showing the correlation scores for offsets in the range $-100 <= x <= 100$. We see a two dip pattern for each of the correlation measures considered in the analysis, one centered around 4 codons upstream (pearson corr. score $= -0.03705$ with a p-value of $2.68e^{-71}$, spearman corr. score of -0.06472, p-value $= 5.25e^{-214}$) and the other one around 1 codon downstream (pearson corr. score $= -0.03398$, p-value $= 1.13e^{-60}$, spearman corr. score of -0.05875 with a p-value of $8.66e^{-178}$). Table 6.2 lists the correlation scores and corresponding p-values at different offsets of the secondary structure energy-window from the footprint read-window, for all three correlation measures. We re-did the analysis considering inverted PARS scores of the RNA sequences instead of

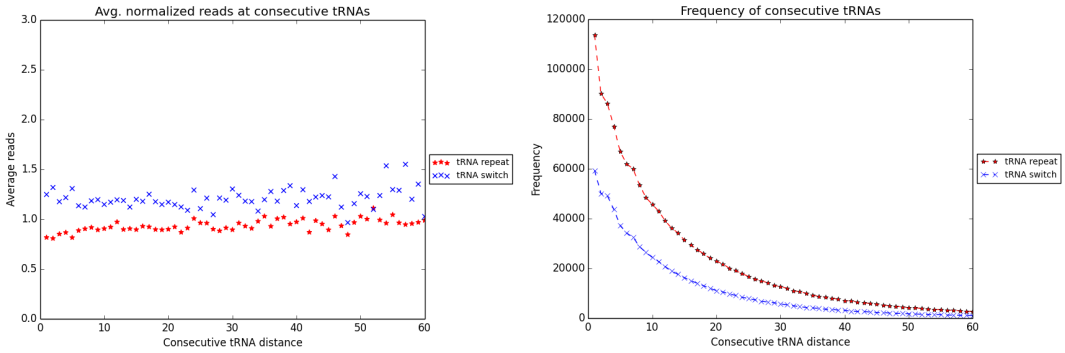

Figure 6.8: Compare the tRNA auto-correlation effect - average reads for tRNA repeat vs tRNA switch. Left: average reads at different tRNA-pair distances. Average reads for cases with tRNA switch (blue crosses) are higher compared to the average reads for cases where tRNA is re-used (red stars). Right: Frequency of tRNA-pairs at different distance ranges.

using the energy of the bonded structures, which confirmed the two dip pattern (see fig. B.3).

To verify the results, we repeated the analysis using in silico random data set and wildtype mRNA-seq data (fig. 6.7, B.4). We did not find any correlation between the structure energy and the reads at the random data-set. Pearson correlation score at 4 codons upstream was 0.001 with a p-value of 0.498 and 1 codon downstream it was 0.002 with a p-value of 0.168 (see supplementary table A.10 for complete statistics). However, for wildtype mRNA-seq data, we have seen a quite similar pattern with less prominent valleys. Pearson correlation score at 4 codons upstream was -0.0157 with a p-value of $1.44e^{-12}$ and 1 codon downstream it was -0.0177 with a p-value of $1.36e^{-15}$ (supplementary table A.11). Similar correlation pattern for both profile data and mRNA-seq data may indicate bias introduced by the data generation method, as the same sequence library preparation method was used to generate both of these data sets, which made it harder to predict the effect of RNA secondary structure on ribosomal pauses at coding regions.

6.3 tRNA auto-correlation effect analysis

mRNA translation is a complex and energy consuming process, and efficiency and accuracy is very important in this process. For better understanding of how translation was shaped and optimized during evolution and how it is affected by any alteration to the natural process, proper knowledge of allocation

Table 6.3: Average reads at 'tRNA repeat' vs 'tRNA switch'

Distance	tRNA repeat	tRNA switch	p-value
1-10	0.855	1.212	0.001
11-20	0.913	1.177	0.001
21-30	0.920	1.163	0.001
31-40	0.964	1.204	0.001
41-50	0.948	1.212	0.001
51-60	0.994	1.278	0.001

and interactions of various resources are important [170].

Several factors contribute to the optimization of the translation process. Among those, codon usage bias effect on translation optimization has been studied extensively [6, 51, 52, 88, 89, 90, 91, 92, 93, 94, 171]. The claim is that, frequent codons improve translation efficiency [8, 158, 171]. However, it is not the only factor to be considered in translation optimization. There are evidences of highly expressed genes having lower codon usage scores [172]. Among other factors, codon pair bias and strength of the folded RNA secondary structures are considered to be important in controlling translation efficiency [8, 12, 13, 142, 146, 147, 150, 151, 167].

Along with optimizing the mRNA sequence, structures and concentrations of various molecules participating on the translation process have been studied widely [52, 54, 126, 129, 152, 154, 155, 173]. At the same time, increasing attention is being given on the movements of the tRNA molecules surrounding an active ribosome [10, 174, 175, 176, 177, 178]. During the course of an active ribosome halting at a codon and then progressing to the next one, availability of the right tRNA at the right moment is critical for efficient and accurate translation. As the cell interior is highly crowded [179], tRNAs may not diffuse away freely through the cytoplasm. Stapulionis et al. suggested a channeled tRNA cycle during protein synthesis in mammalian cells [175]. According to Cannarozzi et al. [10], tRNA re-use plays a significant rule in this optimization process as it minimizes the time to match the proper tRNA. Based on the study on *S. cerevisiae* they found that, autocorrelation improves the translation efficiency significantly. Later, tRNA recycling was proven to be beneficial at the translation mechanism of other organisms as well [177, 178].

However, no study have been performed yet on genome wide tRNA recycling phenomena using ribosome profile data. If re-use of a tRNA takes less time than another already charged tRNA diffusing into the area, then less reads should be observed for the repeat of the tRNAs compared to the situation where a tRNA flip occurs. We have studied the impact of tRNA

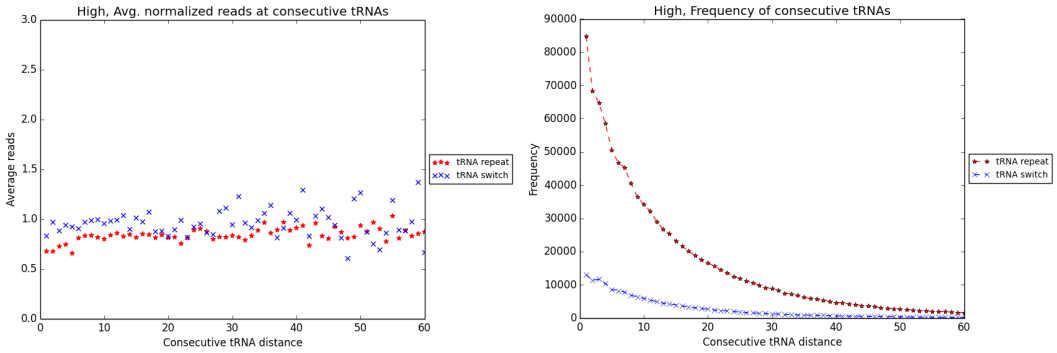

Figure 6.9: Compare the tRNA auto-correlation effect - reads at tRNA repeat vs tRNA switch for pairs of tRNAs with highly abundant codons (codon-usage >= 16.0). Left: read statistics for tRNA repeat vs tRNA switch with highly abundant codons at both tRNA positions. Average reads at tRNA switch cases (blue crosses) are significantly higher compared to the average reads at repeats of the tRNA (red stars). Right: frequency of tRNA-pairs with highly abundant codons at different distance ranges.

auto-correlation on ribosome profile data.

We considered two different situations:

- *Type 1: tRNA repeat vs tRNA switch* – Occurrences of each amino acid having more than one tRNA were divided into two groups: consecutive tRNA repeat vs switch of the tRNA at the next occurrence. If we compare the reads at the second tRNA, we should observe less reads at the repeat of the tRNA compared to the reads at cases where a tRNA switch occurs.

- *Type 2: tRNA repeat distance impact* – For each amino acid, occurrences of 2 consecutive tRNA repeats were considered. Consider a situation where tRNA t_1 repeats twice ($t_1...t_1...t_1$). Let the distance between the first repeat is d_1 and the distance between second repeat is d_2. If $d_1 > d_2$, then we should see less reads at d_2 compared to the reads at d_1, as it would take less time for the ribosome to find the correct tRNA. Conversely, if $d_1 < d_2$, then we should see more reads at d_2. Cases having zero read either at the second or third tRNA position were ignored.

Table 6.4: Average reads at 'tRNA repeat' vs 'tRNA switch' for different codon-usage groups.

Highly abundant codon group:

Distance	tRNA repeat	tRNA switch	p-value
1-10	0.7412	0.9233	0.001
11-20	0.8342	0.9601	0.001
21-30	0.8292	0.9306	0.001
31-40	0.8725	1.0092	0.001
41-50	0.8597	1.0117	0.002
51-60	0.8807	0.9005	0.189

Moderate usage codon group:

Distance	tRNA repeat	tRNA switch	p-value
1-10	1.1647	1.3838	0.001
11-29	1.1773	1.2973	0.279
21-30	1.1422	1.2755	0.012
31-40	1.1449	1.3322	0.289
41-50	1.0621	1.2721	0.076
51-60	1.0978	1.2355	0.443

Under-represented codon group:

Distance	tRNA repeat	tRNA switch	p-value
1-10	1.650	1.999	0.001
11-20	1.403	1.688	0.002
21-30	1.410	1.555	0.001
31-40	1.394	1.588	0.024
41-50	1.376	1.764	0.015
51-60	1.330	1.989	0.003

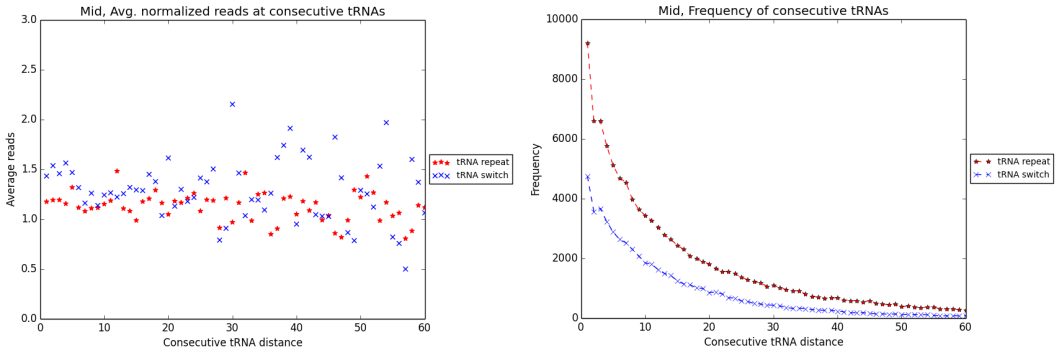

Figure 6.10: Compare the tRNA auto-correlation effect - reads at tRNA repeat vs tRNA switch for pairs of tRNAs with moderately available codons (10.0 <= codon-usage < 16.0). Left: read statistics for tRNA repeat vs tRNA switch. Average reads at tRNA switch cases (blue crosses) are significantly higher compared to the average reads at the repeats of the tRNA (red stars), the effect is maximum till the tRNA pair distance of 10. Right: frequency of tRNA-pairs with moderately available codons at different distance ranges.

6.3.1 Compare average reads at tRNA repeat vs tRNA switch

For each gene, reads were normalized by the average reads at that gene. Top 10% of the genes were ignored, based on the RPKM measure. Fig. 6.8 is showing the read statistics for *type 1* analysis - occurrences of consecutive tRNAs. We would expect auto-correlation effect to be more prominent at a shorter distance. In the left image we see, when tRNA is re-used (red stars), average reads are much lower than the case when a tRNA switch occurs (blue crosses). The right image is showing the frequency of the cases considered at each tRNA-pair distance. Mean of the reads at different distance ranges for both 'tRNA repeat' and 'tRNA switch' and the corresponding p-values of the differences in average reads of these two groups are shown in table 6.3.

Permutation statistics were used to compute the p-values in different buckets (as shown in table 6.4). We randomly swapped average reads at each tRNA pair distance from 'tRNA repeat' and 'tRNA switch' cases. Let a total of N 'tRNA repeat' and 'tRNA switch' cases were considered at a tRNA pair distance D, of which X cases are for 'tRNA repeat' and Y cases are for 'tRNA switch'. To compute the p-value, we randomly assigned X cases from the N available cases to the 'tRNA repeat' group and Y cases to the 'tRNA switch' group. Next, we computed the mean of the reads at different distance buck-

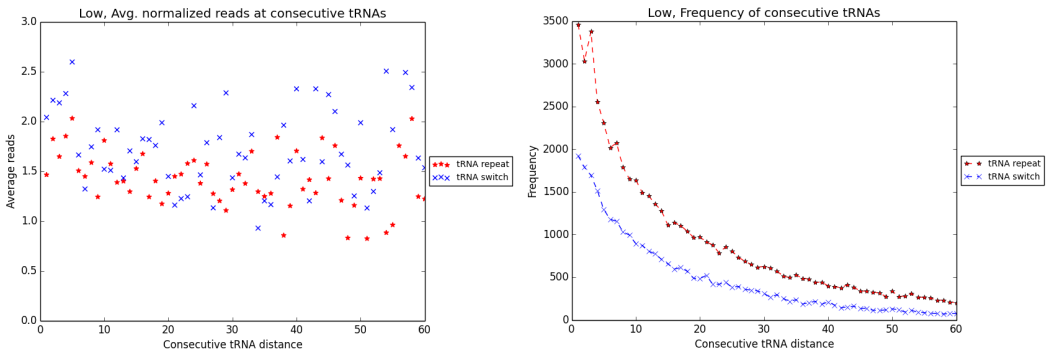

Figure 6.11: Compare the tRNA auto-correlation effect - reads at tRNA repeat vs tRNA switch for pairs of tRNAs with rare codons (codon-usage < 10.0). Left: read statistics for tRNA repeat vs tRNA switch. Average reads at tRNA switch cases (blue crosses) are significantly higher compared to the average reads at repeats of the tRNA (red stars). Right: frequency of tRNA-pairs with rare codons at different distance ranges.

ets and compared the mean with the previously calculated means of 'tRNA repeat' and 'tRNA switch' cases. The whole process was repeated 1000 times to get the final p-values at each distance bucket.

To test whether codon usage bias is affecting the analysis or not, we separated the cases into three different groups. In the first group (high), consecutive occurrences of tRNAs with highly abundant codons at both tRNA positions were considered ($usage >= 16.0$). Pairs of tRNAs with codon-usage in the range of 10.0 to 16.0 were considered in another group (mid) and occurrences with codon-usage less than 10.0 were considered in the third group (low). Fig. 6.9 is showing the tRNA statistics for highly abundant codon-usage group. Fig. 6.10 and 6.11 are showing the statistics for tRNA pairs with moderately available and rare codons at both tRNA positions. In all of these cases we again see that, the average reads for 'tRNA repeat' is lower compared to the average reads for 'tRNA switch'.

Mean read statistics and corresponding p-values at different distance ranges for each of these groups have been shown in table 6.4. Here cases were divided into 6 buckets based on the tRNA repeat/ switch distance, and the average of all the values in that bucket have been considered. We see that, the auto-correlation effect is independent of the codon usage effect. Note from table 6.4, the difference between mean reads is maximum for under-represented codon group. Re-use of the already available tRNA can be particularly beneficial for this group, as finding another rare tRNA would take more time. Also note,

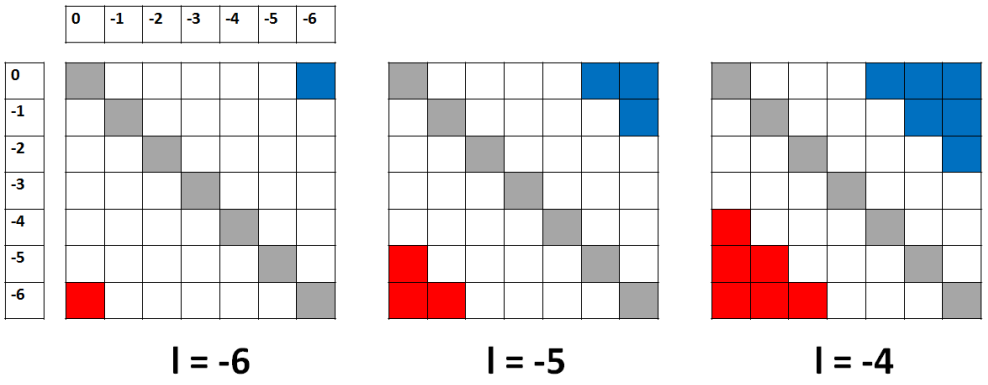

Figure 6.12: Triangles away from the matrix diagonal $M(i, i)$, cases for $l = -6$ at left, for $l = -5$ in the middle and for $l = -4$ at right.

the average reads for both groups with rare codons have been increased by almost 1.5 times from the average reads at highly abundant codon group and auto-correlation effect is also visible.

6.3.2 tRNA repeat distance impact on auto-correlation

Auto-correlation effect decays slowly with distance [10]. To analyze the effect of tRNA pair distance, we considered 2 consecutive tRNA repeats ($t_1...t_1...t_1$) at positions x_0, x_1 and x_2 of an amino acid sequence (*type 2* analysis). Let, $x_1 - x_0 = d_1$ and $x_2 - x_1 = d_2$. The claim is that, if distance d_1 is larger than d_2 and if auto-correlation is effective till distance d_2, then ribosome would spend relatively less time at d_2, and vice versa. Let, reads at x_1 is r_1 and at x_2 is r_2. We considered all cases with both r_1 and r_2 greater than zero. Now, consider a 2-dimensional matrix $M(I, J)$, where $I = max(D_1)$ for all $d_1 \in D_1$ and $J = max(D_2)$ for all $d_2 \in D_2$. Each entry in M has the value $M(d_1, d_2) = \log(\frac{r_2}{r_1})$. If multiple values map into same matrix location, that entry gets the average of all of these values and later in the analysis is weighted by the frequency of occurrence.

If we consider a diagonal along the matrix, negative values should be more common below the diagonal till a certain range where auto-correlation effect is visible. Similarly positive values should be more common above the diagonal. Consider a triangle at the two extreme corners of the matrix M (fig. 6.12). We consider different offset distances. In fig. 6.12, for $l = -6$ only one entry at the corner is considered. For $l = -5$, size of the triangle increases by one more diagonal row toward the main diagonal and so on. We compared the red vs blue intensities at the triangles considered at two extreme corners of the

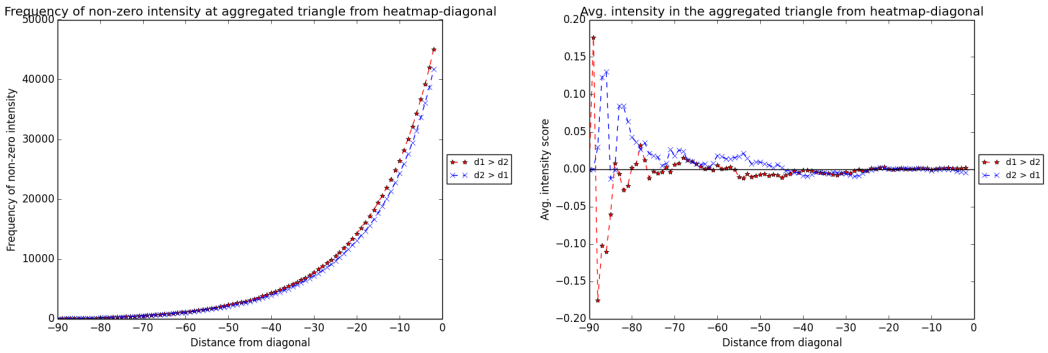

Figure 6.13: Auto-correlation effect analysis at triangles away from the matrix diagonal. Left: Frequency of consecutive tRNA repeats. Right: average of the log-ratios of the reads at the second tRNA repeat vs first tRNA repeat. Red line with stars - distance of the second tRNA repeat is shorter than the distance of the first tRNA repeat. Blue line with crosses - distance of the second tRNA repeat is larger than the distance of the first tRNA repeat.

matrix (fig. 6.13). The intensity scores at these two triangles should be largely different for smaller triangles (blue triangle should have more positive values indicating more reads at the later tRNA and red one should have more negative values indicating more reads at the earlier tRNA). However, as the size of the triangle increases, it will cross the auto-correlation distance boundary and we expect to see no difference in these two curves. We see from the figure that, till approximately around $l = -45$, red and blue curves are showing expected patterns and after that the effect diminishes.

6.4 Discussion

mRNA translation is a complex process which involves active participation of various molecules. It is critical to understand the whole process to control the amount of protein production, to study genetic diseases and to aid in successful vaccine design. Extensive studies are going on to properly understand the whole mechanism. We work on optimizing (or de-optimizing) the steps in mRNA translation process, to aid in controlling the amount of protein production. Hence, we are interested in studying delays introduced by various participating molecules at different steps of the process.

Ribosome profile data gives us a clear picture of the active ribosome positions on a translating mRNA. From the ribosome footprints we can determine approximately how much time ribosome spent at each codon. In an ideal world,

it would spend equal time at each codon. However, we see large variability in the delays it makes while traveling along the sequence (see fig. 5.1, B.1, B.2). Several factors may affect the speed of an active ribosome. We studied the impact of codon usage and found delays introduced by synonymous codons are well-correlated with the usage bias of the codons (chapter 5). Speed of the translation largely depends on the availability of various resources. Optimal use of these resources (such as, tRNA recycling) may aid in faster translation (sec. 6.3). Context surrounding a codon also impacts the translation process (sec. 6.1). We also have worked on determining the impact of RNA secondary structure (sec. 6.2). Amino acid type can also impact the speed of translation. Significantly large peak at the ribosomal P-site was noticed for codons coding for *proline* (see chapter 5 for detail). Presence of the start codon (coding for *methionine*) can also be a factor. Ribosomal pauses are largely determined by any one or combination of these factors.

It is hard to exactly quantify the contribution of each of these factors in ribosomal delays, as it will largely depend on the surrounding environment at which the translation is taking place. Moreover, nature may optimize one factor to compensate for the other. Recent study by Gorochowski et al. showed that, codons belonging to abundant tRNAs are preferentially used in strongly bonded regions and codons read by less abundant tRNAs are used in less structured regions [173]. See appendix B.1 for per codon plot of different factors for yeast genes *YAL038W* and *YKL152C*; read pile-up varies largely with the variation of the factors affecting the translation mechanism.

Contribution of each factor in the ribosomal speed may be correlated in a global manner. We have worked on identifying impact of several known factors affecting the translation process. However, there are issues yet to explore. Once the effect of individual factors have been figured out, it would be interesting to study the combined effect - nature may often trades off one factor to compensate for the effect of another. Detailed understanding of these factors may aid in solving unanswered questions regarding the protein synthesis and its optimization process. Further, it may open up new biological facts yet to explore.

Chapter 7

Conclusion

Synthetic biology is a relatively young field, being popular with days because of its contributions to the modern science. It combines science and engineering together, where the goal is to create new biological systems and functions not available in nature and to find out how life works and how to use it to benefit our society. It can also be defined as engineering technology based on living systems [180]. It's field of application is not limited to biological or medical sciences, but also to take better care of the environment, design modern biosynthesized technology, and to provide better source of nutrition to the growing population.

In 1978, Szybalksi and Skalka wrote about synthetic biology:

The work on restriction nucleases not only permits us easily to construct recombinant DNA molecules and to analyse individual genes, but also has led us into the new era of 'synthetic biology' where not only existing genes are described and analyzed but also new gene arrangements can be constructed and evaluated [181].

The question arises, why engineering of synthetic biology is still expensive and unreliable? According to Drew Endy, either we do not have much knowledge about biological systems or these are to complex to reliably engineer [182]. Alternately, evolution might not have optimized natural biological systems for the purposes of human understanding and engineering [183].

However, the hope is that tremendous improvements are being achieved with time. The cost for large-scale synthesis is dropping rapidly, and is now about \$0.24/*base* for kilo-base sequences. Researchers can now successfully design synthetic viruses [184] and can design vaccines by synthesizing attenuated forms of the viruses [9, 23]. Complex gene circuits are being formed to detect changes in the cancer cells [185]. Novel drugs are being discovered and expensive drugs are being made cheap and easily available through the synthesis of rare natural products [186, 187].

Though synthetic biology is more about doing or building, proper knowledge of what role a particular biological component plays and how various functional components interact with each other is critical. We work on both design and explore phase of synthetic biology. We analyze biological data to better understand the role of different cellular components and their impact on gene translation mechanism. At the design phase we utilize the explored knowledge to optimize or control gene expression.

During the gene translation process, large number of unique functional components interact in various ways. To control the amount of protein production, controlling the gene expression mechanism is essential. Through years of experiments, researchers determined several factors affecting translation process, either by improving or degrading the amount of protein production. One widely accepted factor is codon usage bias. However, there is no direct evidence of how codon usage bias affects the translation process. Through our analysis of ribosome profile data, we showed that more frequent codons (belonging to tRNAs with higher concentration) are translated relatively faster. We further have shown that, GC-rich codons are relatively slow in the translation process. Type of amino acid also plays a role in the translation process. We found *proline* (which has a different chemical structure compared to other amino acids) is slow at the ribosome P-site, where the peptide bond is formed.

We further have worked on analyzing ribosome profile data to evaluate the impact of codon pair bias. The concept of codon pair bias has been utilized by the researchers to design vaccines [9, 23]. Here the idea is to change the relative frequency of the neighboring codons keeping the codon usage frequency same. Some researchers suggested, over-abundant codon pairs translate relatively slowly compared to under-represented codon pairs [151]. Other researchers found, under-represented codon pairs decrease the rate of protein translation [9]. The latter finding was attributed to the compatibilities of adjacent tRNAs at the ribosome [9, 155]. In our analysis, less average reads (valleys) were observed at the codon pair groups under-represented in-frame. Analysis of the codon pair bias effect on ribosome profile data (considering abundance of the pairs in frame vs out of frame) revealed a reversed pattern compared to that for codon usage bias. This may indicate faster ribosomal transaction at the under-represented codon pairs. On the other hand, codon pairs over abundant in all coding frames also showed a valley at position 6 of the read window. An alternate hypothesis of the effect of codon pair bias could be attributed to the frame-shifting event. Probably the codon pairs causing more frame-shifts are under-represented in-frame to avoid translation error, while over-represented out of frame to overcome a shift of the ribosome from the coding frame. Further evidence is needed to validate any of theses hypotheses.

The effect of secondary structure on message translation has been studied by researchers for years [13, 15, 16, 17, 18, 167]. Some researchers have suggested that, secondary structure is preferred at the 5' end of the mRNA [8, 13]. Katz et al. found statistically significant correlation between natural mRNAs and corresponding local secondary structures [168]. We worked on determining the per codon effect of the secondary structures at the coding regions of an mRNA on an active ribosome. We found significant correlation at around 4 codons upstream and 1 codon downstream of the active ribosome position. We did not see any such correlation for in silico random data. However, we have seen quite similar correlation to a lesser extent for wildtype mRNA-seq data. Hence we could not come up with a conclusion about the direct impact of folded RNA secondary structure on ribosome speed. Part of the impact could be due to the bias introduced by the sequence library preparation method.

We also have worked on predicting tRNA auto-correlation impact on ribosome profile data. Several in vivo experiments conducted on different organisms found recycling of tRNAs beneficial for the translation process [10, 177, 178]. However, no genome wide study of the impact have been performed yet. We analyzed high coverage ribosome profile data on 4801 *Saccharomyces cerevisiae* genes. Our analysis results show that, on average reads at the positions where a tRNA switch occurs are comparatively higher than at the positions where previously used tRNA is repeated. Moreover, distance is also important in this recycling process. Re-use of the tRNA at a shorter distance results in a lower average reads compared to a repeat further apart.

The method of ribosome profiling has opened up new ways in front of the researchers. As the ribosome is at the heart of the translation process, detailed knowledge of where it pauses and how long it spends at each site during the translation process will give us clear answers to many debated questions and at the same time will guide us to factors not known yet. We worked on predicting the impact of several factors. Interesting results came out of the analysis, which may aid in clarifying some widely accepted concepts in the translation mechanism. Our lab is further working on the project to get deeper understanding of the whole mechanism.

We also have worked on statistical analysis of microarray data, which enables to research on expression changes on a large set of genes. Through the analysis of the same set of genes at different experimental conditions we can broadly determine the behavior of the genes, and how it is impacted by the environmental changes inside the cell. We worked on identifying housekeeping genes and on predicting differentially expressed genes through the analysis of *Murine Gammaherpervirus 68* microarray data. Interesting facts about virus life cycle came out of the analysis. Enrichment analysis of the differentially

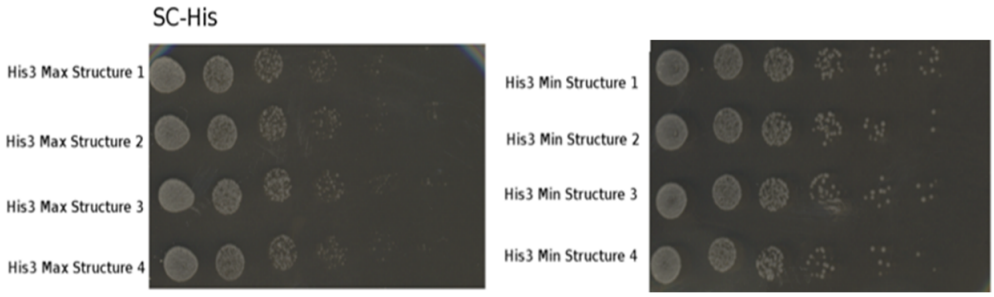

Figure 7.1: Laboratory synthesis of energy optimized/ de-optimized yeast gene *YOR202W*. Each row represents independent replicates of the designed max or min structure sequence strains. Each column has different number of cells per spot, 50000 cells per spot for the first column, following columns represent serial three fold dilutions of the earlier columns. (Synthesized by: Justin Gardin, B. Futcher Lab.)

expressed gene-sets would be interesting to get better understanding of the functional profile of those gene-sets.

Along with analyzing biological data, we work on developing novel algorithms to design optimized genes. We developed algorithm to design max structure (min energy) and min structure (max energy) RNA sequences. Folding energy is an important factor in determining translation efficiency [21]. The amount of secondary structure may aid in controlling the amount of protein production from a given gene [8, 13]. Based on the laboratory synthesis of our designed sequences, the amount of protein formed for max structured sequence was less compared to the min structured sequence (fig. 7.1).

We also have worked on designing genes with maximum and minimum tRNA auto-correlation (chapter 3). In our work, we have proposed a new distance dependent measure of tRNA sequence auto-correlation (DICA), while previous studies focused only on the number of tRNA changes [47, 48]. Using our algorithms we have designed 5018 auto-correlation optimized yeast genes considering distance between pairs of tRNAs into account.

Factors studied so far in our analysis cover only parts of the complex biological processes involved in protein translation mechanism. We mostly covered factors involved in the translation elongation step. However, overall translation mechanism is significantly impacted by the initiation step [8]. Detailed study of the factors involved in translation initiation and their impact on ribosome profile data remains as an important future challenge.

Appendix A

Supplementary tables

A.1 Measurement of average decoding rates of the 61 sense codons in vivo

Complete ribosome residence times for each codon at each of the 10 possible codon positions in a 30 nt (or, for Ingolia data, 24 nt) ribosome footprint are shown in table A.1, A.2, A.3, A.4 and A.5. Each table is based on data from an independent biological experiment. Four of these experiments were done during the course of this work, two experiments by JG and two experiments by YC, while the fifth experiment was published by Ingolia et al. [85].

Table A.6, A.7 and A.8 are showing complete Ribosome Residence Times for each codon at each of the 7 possible codon positions in a 21 nt ribosome footprint. Each table is based on one of the three anisomycin datasets of Lareau et al. [87].

A.2 Codon pair bias effect analysis on ribosome profile data

Weighted mean RRT of the *high* vs *low* codon pair groups have been listed in table A.9. Three different scoring methods have been used for measuring the abundance of a codon pair: classic, gap and signal.

Table A.1: Ribosome residence time analysis for all codons from the SC-lys expt.

Codon	*Pos*1	*Pos*2	*Pos*3	*Pos*4	*Pos*5	*Pos*6	*Pos*7	*Pos*8	*Pos*9	*Pos*10
CTC	0.0944	0.0956	0.0969	0.1063	0.0667	0.1892	0.0997	0.0932	0.0785	0.0798
CCC	0.0804	0.0879	0.0799	0.1089	0.1479	0.1715	0.0991	0.0866	0.0665	0.0712
GGG	0.1098	0.1003	0.0915	0.0810	0.0957	0.1609	0.0846	0.0840	0.1115	0.0808
AGG	0.1082	0.0720	0.0865	0.0875	0.0706	0.1595	0.0979	0.0873	0.1163	0.1143
ATA	0.0908	0.0882	0.1114	0.1106	0.0797	0.1567	0.1042	0.0887	0.0957	0.0741
GGA	0.0835	0.1083	0.0929	0.0794	0.1095	0.1558	0.0891	0.1003	0.0920	0.0892
TGG	0.1291	0.0892	0.0844	0.0945	0.0720	0.1526	0.0861	0.0852	0.1223	0.0846
GTG	0.1004	0.0972	0.0869	0.0782	0.0753	0.1519	0.1018	0.0979	0.1135	0.0970
CGC	0.1194	0.1054	0.1037	0.0883	0.0757	0.1453	0.0884	0.0901	0.0924	0.0912
CGA	0.0897	0.0912	0.0763	0.0882	0.1146	0.1447	0.1136	0.1133	0.0936	0.0747
CGG	0.1171	0.0745	0.0637	0.0854	0.1345	0.1436	0.0773	0.0839	0.1228	0.0972
TCG	0.1062	0.0875	0.0952	0.0909	0.0787	0.1427	0.1045	0.1019	0.1172	0.0752
CCA	0.0962	0.0888	0.0822	0.1145	0.1477	0.1382	0.0936	0.0914	0.0896	0.0578
ACA	0.0813	0.0874	0.0898	0.1044	0.1044	0.1347	0.0998	0.1117	0.1098	0.0766
CCG	0.1109	0.0839	0.0719	0.1150	0.1302	0.1312	0.0986	0.0808	0.0872	0.0904
GTA	0.0764	0.1025	0.1049	0.0962	0.1136	0.1306	0.0987	0.0939	0.0996	0.0835
GCA	0.0972	0.1074	0.0913	0.0982	0.1029	0.1285	0.1033	0.1033	0.0888	0.0790
CCT	0.0999	0.0914	0.0811	0.1101	0.1801	0.1271	0.0934	0.0840	0.0785	0.0543
TCA	0.1020	0.0994	0.1058	0.1058	0.1021	0.1264	0.0987	0.1013	0.0907	0.0677
TAC	0.1057	0.0837	0.0942	0.1040	0.0844	0.1252	0.1084	0.1032	0.0902	0.1010
TAT	0.0960	0.0852	0.1092	0.1169	0.1042	0.1251	0.1026	0.0941	0.1027	0.0641
GAG	0.0930	0.0922	0.0933	0.0851	0.0780	0.1247	0.0988	0.1030	0.1102	0.1218
CTA	0.0971	0.0943	0.1090	0.1216	0.0964	0.1246	0.1022	0.0947	0.0974	0.0627
CTT	0.1052	0.0938	0.1048	0.1277	0.0866	0.1238	0.0980	0.0998	0.0971	0.0632
TGC	0.1063	0.0958	0.1005	0.0858	0.0938	0.1228	0.0917	0.0991	0.0853	0.1189
GGC	0.1029	0.1075	0.0976	0.0918	0.1091	0.1215	0.0876	0.0956	0.0783	0.1081
CAG	0.1187	0.0786	0.0883	0.0858	0.0999	0.1151	0.1011	0.1034	0.1106	0.0985
ACG	0.1137	0.0920	0.0823	0.1006	0.0951	0.1116	0.1022	0.1053	0.0886	0.1086
AGT	0.0989	0.0872	0.0986	0.0972	0.0801	0.1104	0.1074	0.1059	0.1077	0.1067
AGC	0.0940	0.0830	0.0982	0.1176	0.0707	0.1092	0.0998	0.0944	0.0894	0.1438
CAC	0.1094	0.0911	0.0960	0.1015	0.0975	0.1082	0.1078	0.1092	0.0879	0.0915
TTT	0.0947	0.0778	0.1083	0.1088	0.1046	0.1048	0.1047	0.1060	0.1159	0.0743
GAA	0.0737	0.1114	0.1041	0.0908	0.0917	0.1041	0.0934	0.0983	0.0953	0.1373
AGA	0.0932	0.0826	0.0897	0.0851	0.0900	0.1013	0.1069	0.1038	0.1139	0.1335
TTC	0.1058	0.0903	0.1030	0.0956	0.0940	0.1000	0.1013	0.1025	0.0871	0.1204
GCG	0.1090	0.1107	0.0931	0.1030	0.0775	0.0995	0.1008	0.0993	0.0976	0.1095
TCC	0.1170	0.1047	0.1012	0.0942	0.0788	0.0989	0.1017	0.0993	0.0848	0.1195
TTA	0.0944	0.0823	0.1105	0.1185	0.1075	0.0985	0.1059	0.1028	0.1102	0.0693
TCT	0.1067	0.1028	0.1073	0.1021	0.1014	0.0981	0.1089	0.0985	0.0964	0.0777
CAT	0.0971	0.0943	0.1062	0.1139	0.1290	0.0930	0.1093	0.1000	0.0968	0.0604
GGT	0.1077	0.1290	0.1065	0.0963	0.1192	0.0926	0.0842	0.0940	0.1000	0.0705
ATG	0.1231	0.0854	0.0986	0.0934	0.0812	0.0923	0.1090	0.1022	0.1173	0.0975
ATT	0.0886	0.0894	0.1130	0.1203	0.1057	0.0922	0.1041	0.0992	0.1063	0.0813
TTG	0.1289	0.0917	0.0981	0.1074	0.0809	0.0920	0.0991	0.0982	0.1180	0.0857
CTG	0.1289	0.0919	0.0923	0.1161	0.0847	0.0916	0.1136	0.0894	0.1116	0.0798
AAT	0.0823	0.0851	0.1070	0.1073	0.1389	0.0879	0.1041	0.0896	0.0955	0.1023
AAA	0.0659	0.0797	0.1044	0.0937	0.1059	0.0878	0.1050	0.0909	0.0946	0.1720
CGT	0.1060	0.0892	0.0969	0.0925	0.0946	0.0875	0.1059	0.1108	0.1258	0.0908
CAA	0.0924	0.0954	0.1034	0.0994	0.1058	0.0872	0.1124	0.1012	0.1134	0.0894
GCC	0.1120	0.1226	0.0965	0.0996	0.0935	0.0861	0.1024	0.1028	0.0756	0.1090
GAC	0.0881	0.1171	0.1048	0.0932	0.0999	0.0851	0.0922	0.1120	0.0884	0.1192
TGT	0.1181	0.0909	0.0967	0.0904	0.1049	0.0813	0.1045	0.1017	0.1179	0.0936
GCT	0.1034	0.1274	0.0999	0.1041	0.1025	0.0809	0.0993	0.1041	0.0902	0.0880
ATC	0.1007	0.1077	0.1053	0.1050	0.0907	0.0804	0.0973	0.1033	0.0836	0.1259
ACT	0.0934	0.1056	0.1009	0.1174	0.1088	0.0778	0.1043	0.1062	0.0999	0.0857
GAT	0.0900	0.1275	0.1187	0.0959	0.1177	0.0757	0.0900	0.1050	0.0957	0.0839
AAC	0.0886	0.0823	0.0968	0.0978	0.1180	0.0756	0.1007	0.1023	0.0860	0.1518
GTT	0.0971	0.1200	0.1181	0.1030	0.0867	0.0754	0.1000	0.1050	0.1088	0.0858
GTC	0.1009	0.1190	0.1112	0.0965	0.0813	0.0754	0.1046	0.1019	0.0900	0.1191
AAG	0.0958	0.0763	0.1046	0.0920	0.0937	0.0741	0.1076	0.0956	0.1130	0.1473
ACC	0.1127	0.1094	0.1011	0.1109	0.0914	0.0697	0.1090	0.1043	0.0825	0.1090

Table A.2: Ribosome residence time analysis from the YPD1 (WT) expt.

Codon	*Pos*1	*Pos*2	*Pos*3	*Pos*4	*Pos*5	*Pos*6	*Pos*7	*Pos*8	*Pos*9	*Pos*10
TGG	0.0922	0.0831	0.0793	0.1302	0.1126	0.1465	0.0818	0.0726	0.1127	0.0890
GGG	0.0996	0.0919	0.0826	0.0951	0.1004	0.1454	0.0800	0.0802	0.1070	0.1178
GGA	0.0876	0.1060	0.0883	0.0888	0.1232	0.1415	0.0854	0.0923	0.0959	0.0909
GAG	0.0957	0.1041	0.0838	0.0828	0.0819	0.1415	0.0874	0.1012	0.1226	0.0990
TAC	0.0874	0.0929	0.0984	0.1035	0.0946	0.1393	0.1056	0.1007	0.0975	0.0803
GGC	0.1079	0.1179	0.0997	0.0850	0.1058	0.1388	0.0798	0.0878	0.0714	0.1060
CCG	0.1161	0.0915	0.0678	0.1055	0.1534	0.1248	0.0901	0.0928	0.0853	0.0726
CCA	0.0967	0.0911	0.0883	0.1145	0.1574	0.1228	0.0984	0.0877	0.0875	0.0556
TCG	0.0951	0.0929	0.0897	0.1192	0.0852	0.1207	0.1021	0.1055	0.1137	0.0759
AGG	0.1150	0.0723	0.0929	0.1157	0.0650	0.1198	0.0863	0.0887	0.1157	0.1286
CGC	0.1372	0.1080	0.1183	0.1028	0.0807	0.1189	0.0914	0.0778	0.0844	0.0805
GAA	0.0824	0.1243	0.1005	0.0990	0.0781	0.1183	0.0905	0.0954	0.0977	0.1137
CCC	0.0961	0.0925	0.0870	0.1106	0.1698	0.1179	0.0955	0.0837	0.0747	0.0723
CTC	0.1077	0.1047	0.1163	0.1224	0.0809	0.1171	0.1044	0.0913	0.0795	0.0756
ACA	0.0859	0.0898	0.0967	0.1107	0.0950	0.1159	0.1052	0.1052	0.1041	0.0914
ATA	0.0926	0.0891	0.1062	0.1316	0.0765	0.1156	0.1045	0.0982	0.1000	0.0858
TCA	0.0749	0.0971	0.1074	0.1199	0.0909	0.1149	0.1110	0.1103	0.1089	0.0646
CTA	0.1042	0.0991	0.1123	0.1025	0.1069	0.1130	0.1106	0.0932	0.0970	0.0612
GTG	0.1040	0.1027	0.0914	0.0942	0.0928	0.1125	0.0764	0.0801	0.1011	0.1449
TAT	0.0890	0.0937	0.1084	0.1039	0.1040	0.1114	0.1161	0.1077	0.1075	0.0584
GTA	0.0932	0.1156	0.1162	0.1028	0.0784	0.1111	0.0927	0.0951	0.1028	0.0919
GCA	0.0911	0.1177	0.1149	0.1273	0.0942	0.1106	0.0874	0.0871	0.0869	0.0828
TCC	0.1035	0.1060	0.0968	0.0983	0.0867	0.1106	0.1111	0.1037	0.0907	0.0927
AGC	0.1212	0.0837	0.1013	0.1091	0.0726	0.1097	0.0845	0.0903	0.0839	0.1440
CTG	0.1223	0.0867	0.0878	0.1086	0.1108	0.1087	0.1008	0.0820	0.1046	0.0877
TGC	0.0946	0.1023	0.1024	0.0866	0.1095	0.1063	0.0986	0.1006	0.0921	0.1069
TCT	0.0909	0.1033	0.1118	0.1017	0.1016	0.1061	0.1180	0.1057	0.0919	0.0690
GGT	0.1101	0.1205	0.1043	0.0863	0.1167	0.1051	0.0830	0.0877	0.0904	0.0960
CAG	0.1136	0.0772	0.0832	0.0956	0.1041	0.1030	0.1088	0.1002	0.1166	0.0978
AGT	0.1163	0.0815	0.0988	0.0985	0.0856	0.1020	0.0978	0.0998	0.1075	0.1121
CTT	0.1083	0.0924	0.1098	0.1086	0.1028	0.1017	0.1151	0.1068	0.0921	0.0624
TTA	0.0680	0.0959	0.1115	0.1219	0.0853	0.1015	0.1240	0.1159	0.1214	0.0546
CCT	0.1071	0.0904	0.0850	0.1047	0.1884	0.1005	0.0993	0.0877	0.0747	0.0621
GAC	0.0972	0.1279	0.0978	0.0821	0.0883	0.0996	0.0881	0.1064	0.0995	0.1130
CGA	0.1202	0.1031	0.1207	0.1647	0.1023	0.0982	0.0900	0.0723	0.0692	0.0594
TTG	0.0864	0.0871	0.0913	0.1020	0.0998	0.0976	0.1119	0.1075	0.1309	0.0855
GCC	0.1090	0.1159	0.1014	0.1060	0.0888	0.0956	0.0962	0.0943	0.0758	0.1171
CAC	0.1203	0.0942	0.1010	0.1121	0.1322	0.0941	0.1012	0.0950	0.0811	0.0688
GCG	0.1122	0.1118	0.1011	0.1306	0.0754	0.0939	0.0840	0.0877	0.0880	0.1153
ACG	0.1094	0.0922	0.0761	0.0915	0.1128	0.0933	0.0973	0.1006	0.0964	0.1304
AGA	0.1118	0.0829	0.0935	0.0833	0.0812	0.0931	0.0999	0.1152	0.1220	0.1171
GCT	0.0995	0.1248	0.1113	0.0979	0.1000	0.0898	0.0919	0.0938	0.0837	0.1074
GAT	0.1071	0.1313	0.1114	0.0841	0.1031	0.0894	0.0906	0.1052	0.0929	0.0850
TTC	0.0847	0.0979	0.1058	0.1094	0.0944	0.0887	0.1164	0.1092	0.1003	0.0932
AAA	0.0876	0.0858	0.0973	0.0898	0.1055	0.0874	0.0998	0.0970	0.0959	0.1540
CAA	0.1106	0.0881	0.0989	0.1020	0.1049	0.0870	0.1154	0.1058	0.1131	0.0743
ATT	0.0958	0.0852	0.1114	0.1079	0.0981	0.0863	0.1115	0.1120	0.1014	0.0904
CGG	0.1052	0.0786	0.0793	0.1144	0.2180	0.0861	0.0790	0.0582	0.0930	0.0882
TTT	0.0783	0.0917	0.1156	0.1102	0.0996	0.0861	0.1215	0.1225	0.1117	0.0628
GTC	0.1149	0.1235	0.1185	0.1059	0.0640	0.0854	0.0944	0.0978	0.0868	0.1089
GTT	0.1091	0.1139	0.1226	0.1041	0.0714	0.0852	0.0990	0.0990	0.0994	0.0963
AAT	0.1020	0.0847	0.0953	0.0888	0.1157	0.0822	0.1077	0.1086	0.1038	0.1111
CAT	0.1102	0.0915	0.1104	0.0983	0.1401	0.0821	0.1095	0.1042	0.0952	0.0585
ATC	0.1116	0.1039	0.1024	0.1088	0.0871	0.0803	0.1031	0.1009	0.0847	0.1172
ACT	0.0956	0.1008	0.1050	0.0943	0.1051	0.0800	0.1078	0.1095	0.0944	0.1074
AAC	0.1066	0.0838	0.0904	0.0865	0.1095	0.0796	0.1032	0.1026	0.0883	0.1495
ACC	0.1193	0.0984	0.0976	0.0996	0.0862	0.0774	0.1154	0.0988	0.0877	0.1197
ATG	0.1221	0.0845	0.0872	0.1060	0.0831	0.0764	0.1004	0.1003	0.1192	0.1210
CGT	0.1271	0.1027	0.1049	0.0845	0.0857	0.0756	0.0995	0.0973	0.1197	0.1030
AAG	0.1045	0.0766	0.0853	0.0874	0.1061	0.0752	0.0993	0.0978	0.1168	0.1510
TGT	0.1050	0.0930	0.0998	0.0812	0.1143	0.0734	0.1074	0.1026	0.1253	0.0980

Table A.3: Ribosome residence time analysis from the YPD2 (whi3) expt.

Codon	*Pos*1	*Pos*2	*Pos*3	*Pos*4	*Pos*5	*Pos*6	*Pos*7	*Pos*8	*Pos*9	*Pos*10
GGG	0.1068	0.1035	0.0785	0.0934	0.1025	0.1385	0.0789	0.0831	0.1197	0.0952
GGC	0.1135	0.1278	0.0962	0.0880	0.1066	0.1385	0.0784	0.0802	0.0760	0.0948
GGA	0.0827	0.1115	0.0891	0.0851	0.1263	0.1374	0.0875	0.0870	0.1005	0.0928
TAC	0.0886	0.0847	0.0986	0.1083	0.1004	0.1363	0.1077	0.1007	0.0903	0.0842
GAG	0.0928	0.1029	0.0890	0.0838	0.0797	0.1348	0.0935	0.1025	0.1273	0.0937
TGG	0.1009	0.0735	0.0813	0.1301	0.1139	0.1302	0.0856	0.0778	0.1272	0.0796
TCG	0.1052	0.0888	0.0872	0.1221	0.0894	0.1273	0.1028	0.0977	0.1108	0.0688
CCA	0.1034	0.0903	0.0856	0.1109	0.1603	0.1249	0.0969	0.0841	0.0825	0.0612
CTC	0.1139	0.1064	0.1153	0.1138	0.0882	0.1225	0.0932	0.0857	0.0775	0.0836
CCC	0.1015	0.0966	0.0889	0.1027	0.1777	0.1211	0.0887	0.0749	0.0708	0.0769
CCG	0.1266	0.0936	0.0664	0.1119	0.1578	0.1197	0.0852	0.0902	0.0807	0.0678
CGC	0.1682	0.1100	0.1185	0.0888	0.0950	0.1188	0.0701	0.0692	0.0786	0.0828
AGG	0.1173	0.0685	0.0949	0.1149	0.0762	0.1184	0.0814	0.0842	0.1312	0.1131
GAA	0.0750	0.1334	0.1001	0.1006	0.0789	0.1149	0.0900	0.0955	0.0932	0.1185
TCT	0.0863	0.0993	0.1139	0.1035	0.1060	0.1144	0.1196	0.1059	0.0854	0.0657
TCA	0.0723	0.0921	0.1075	0.1204	0.0999	0.1133	0.1168	0.1101	0.0972	0.0705
TCC	0.1003	0.1041	0.0942	0.0990	0.0909	0.1127	0.1107	0.1015	0.0871	0.0995
ACA	0.0850	0.0867	0.0974	0.1058	0.0905	0.1126	0.1015	0.1109	0.1066	0.1031
TAT	0.0888	0.0897	0.1113	0.1116	0.1010	0.1113	0.1213	0.1094	0.1006	0.0550
AGC	0.1258	0.0736	0.1046	0.1089	0.0772	0.1113	0.0848	0.0851	0.0909	0.1377
CTT	0.1023	0.0861	0.1132	0.1082	0.1123	0.1095	0.1077	0.1070	0.0904	0.0632
CCT	0.1099	0.0895	0.0839	0.1055	0.1919	0.1079	0.0952	0.0821	0.0708	0.0633
ATA	0.0877	0.0847	0.1113	0.1303	0.0756	0.1075	0.0986	0.1076	0.1023	0.0944
GTG	0.1095	0.1029	0.0938	0.0947	0.0881	0.1071	0.0779	0.0837	0.1035	0.1389
CGA	0.1174	0.0782	0.1099	0.1485	0.1071	0.1050	0.0890	0.0969	0.0792	0.0688
GGT	0.1089	0.1367	0.1055	0.0897	0.1090	0.1050	0.0827	0.0812	0.0950	0.0864
GCA	0.0950	0.1250	0.1145	0.1270	0.0934	0.1048	0.0879	0.0860	0.0803	0.0862
CAG	0.1155	0.0734	0.0810	0.0918	0.1033	0.1039	0.1122	0.1015	0.1213	0.0963
CTA	0.1128	0.0978	0.1124	0.0986	0.1001	0.1014	0.1052	0.1053	0.0942	0.0721
TGC	0.0870	0.0916	0.1092	0.0953	0.1187	0.1013	0.1037	0.0949	0.0974	0.1009
AGT	0.1171	0.0766	0.0986	0.1008	0.0852	0.1012	0.1016	0.1007	0.1163	0.1018
TTG	0.0839	0.0799	0.0914	0.1064	0.1004	0.1007	0.1118	0.1130	0.1317	0.0808
TTA	0.0645	0.0865	0.1152	0.1265	0.0872	0.1002	0.1280	0.1212	0.1169	0.0538
GTA	0.0828	0.1209	0.1179	0.1015	0.0761	0.1000	0.1005	0.0999	0.1028	0.0976
CTG	0.1425	0.0872	0.0860	0.1089	0.1079	0.0996	0.0957	0.0794	0.1095	0.0834
ATT	0.0906	0.0812	0.1149	0.1125	0.0961	0.0932	0.1101	0.1149	0.1038	0.0829
GCC	0.1115	0.1247	0.0969	0.1108	0.0947	0.0931	0.0930	0.0831	0.0742	0.1179
GAC	0.0932	0.1383	0.0933	0.0816	0.0920	0.0926	0.0880	0.1079	0.0988	0.1143
GAT	0.0977	0.1404	0.1086	0.0852	0.1020	0.0916	0.0925	0.1064	0.0963	0.0792
GCT	0.0939	0.1312	0.1145	0.1003	0.0989	0.0914	0.0932	0.0887	0.0791	0.1089
GCG	0.1194	0.1172	0.1005	0.1295	0.0800	0.0911	0.0777	0.0813	0.0933	0.1102
CAC	0.1256	0.0909	0.1013	0.1090	0.1308	0.0908	0.1001	0.0968	0.0784	0.0763
AGA	0.1147	0.0741	0.0927	0.0794	0.0817	0.0902	0.1041	0.1136	0.1343	0.1152
AAT	0.0980	0.0757	0.1031	0.0893	0.1137	0.0893	0.1170	0.1076	0.1035	0.1027
TTT	0.0756	0.0854	0.1202	0.1153	0.0955	0.0888	0.1246	0.1287	0.1077	0.0581
TTC	0.0840	0.0935	0.1051	0.1126	0.0926	0.0888	0.1175	0.1136	0.0936	0.0986
AAA	0.0793	0.0780	0.0990	0.0909	0.1001	0.0887	0.1043	0.1026	0.0945	0.1628
ACG	0.1087	0.0842	0.0800	0.0903	0.1051	0.0886	0.0979	0.1106	0.1011	0.1334
CAA	0.1151	0.0792	0.1008	0.1020	0.1021	0.0866	0.1179	0.1111	0.1078	0.0773
GTC	0.1121	0.1322	0.1131	0.1069	0.0702	0.0861	0.0918	0.0912	0.0844	0.1121
ATC	0.1053	0.0999	0.1008	0.1108	0.0899	0.0860	0.1013	0.1049	0.0829	0.1182
GTT	0.0984	0.1237	0.1295	0.1098	0.0685	0.0854	0.0983	0.0999	0.0993	0.0871
CGG	0.1431	0.0760	0.0714	0.1005	0.2316	0.0849	0.0775	0.0567	0.0802	0.0782
ACT	0.0926	0.0961	0.1059	0.0978	0.1051	0.0848	0.1077	0.1099	0.0925	0.1077
CAT	0.1137	0.0859	0.1125	0.0984	0.1372	0.0826	0.1153	0.1071	0.0876	0.0598
AAC	0.1084	0.0787	0.0893	0.0866	0.1087	0.0817	0.0992	0.1012	0.0866	0.1596
ACC	0.1260	0.0953	0.0938	0.0966	0.0868	0.0811	0.1122	0.0985	0.0855	0.1241
AAG	0.1044	0.0692	0.0874	0.0872	0.1083	0.0797	0.0984	0.1001	0.1228	0.1424
ATG	0.1257	0.0823	0.0916	0.1044	0.0791	0.0795	0.0945	0.1058	0.1255	0.1117
TGT	0.0981	0.0850	0.1018	0.0905	0.1139	0.0791	0.1123	0.0984	0.1285	0.0925
CGT	0.1335	0.0880	0.1065	0.0851	0.0833	0.0784	0.0981	0.0938	0.1278	0.1056

Table A.4: Ribosome residence time analysis from the SC-his expt.

Codon	*Pos*1	*Pos*2	*Pos*3	*Pos*4	*Pos*5	*Pos*6	*Pos*7	*Pos*8	*Pos*9	*Pos*10
GGA	0.1092	0.0910	0.0829	0.0663	0.1255	0.1474	0.1101	0.0794	0.0962	0.0920
GGG	0.1553	0.0958	0.0682	0.0683	0.1200	0.1391	0.0819	0.0654	0.1247	0.0812
CCT	0.0681	0.1189	0.0908	0.0678	0.1820	0.1382	0.1017	0.0878	0.0699	0.0749
CCA	0.0774	0.0902	0.0868	0.0765	0.1634	0.1367	0.1186	0.0911	0.0826	0.0765
CCC	0.0664	0.1274	0.0896	0.0662	0.2120	0.1317	0.0760	0.0981	0.0527	0.0799
CCG	0.0956	0.0927	0.0637	0.0865	0.1927	0.1304	0.0763	0.0994	0.0954	0.0673
CTC	0.0593	0.1329	0.1301	0.1444	0.0611	0.1247	0.0983	0.1040	0.0696	0.0756
AGG	0.1268	0.0764	0.0935	0.0918	0.0756	0.1229	0.1086	0.0907	0.1294	0.0843
CGC	0.0478	0.1551	0.1296	0.1139	0.0718	0.1227	0.1111	0.0986	0.0911	0.0582
GAA	0.0770	0.1035	0.0793	0.0849	0.0839	0.1195	0.1142	0.1022	0.0939	0.1417
TTG	0.1169	0.0629	0.0913	0.1326	0.1062	0.1174	0.0954	0.0875	0.1226	0.0672
GGC	0.1529	0.1282	0.0908	0.0901	0.0977	0.1168	0.0855	0.0654	0.0810	0.0917
CTG	0.1226	0.0724	0.0977	0.1067	0.1052	0.1162	0.0735	0.0808	0.1521	0.0727
TAT	0.0663	0.0770	0.1358	0.1251	0.0988	0.1154	0.1297	0.0997	0.0914	0.0608
CAG	0.1122	0.0803	0.0914	0.1007	0.1352	0.1150	0.0874	0.0850	0.1164	0.0764
AAT	0.0699	0.0664	0.0952	0.0761	0.1240	0.1144	0.1318	0.1197	0.1052	0.0974
TGC	0.1117	0.0768	0.1038	0.1137	0.1421	0.1140	0.0692	0.0985	0.0832	0.0871
CTT	0.0668	0.1030	0.1152	0.1228	0.1041	0.1135	0.1020	0.1172	0.0858	0.0695
TAC	0.0890	0.0926	0.1016	0.1024	0.0871	0.1119	0.1234	0.1231	0.0821	0.0869
GAG	0.1423	0.0798	0.0689	0.0800	0.0887	0.1117	0.0872	0.1016	0.1300	0.1098
GTA	0.0763	0.1224	0.1432	0.1204	0.0738	0.1110	0.0868	0.0801	0.0970	0.0891
CAA	0.0693	0.0836	0.1064	0.0910	0.1315	0.1087	0.1192	0.1331	0.0826	0.0745
AAA	0.0736	0.0753	0.0973	0.1036	0.1059	0.1081	0.1341	0.0984	0.0839	0.1198
TCG	0.0735	0.0832	0.1217	0.1288	0.0650	0.1062	0.1266	0.1132	0.1029	0.0788
GCC	0.1122	0.1433	0.0831	0.0877	0.1023	0.1061	0.1043	0.0820	0.0767	0.1023
AAC	0.0862	0.0797	0.0927	0.0850	0.1179	0.1052	0.1122	0.1179	0.0919	0.1114
TCT	0.0642	0.1021	0.1228	0.1052	0.0816	0.1050	0.1273	0.1190	0.0947	0.0780
CTA	0.0907	0.0989	0.1445	0.1205	0.0862	0.1048	0.0997	0.1216	0.0646	0.0685
CAT	0.0560	0.1016	0.1173	0.1152	0.1238	0.1029	0.1164	0.1154	0.0769	0.0745
GCT	0.0919	0.1444	0.1030	0.0871	0.1093	0.1025	0.0845	0.0905	0.0771	0.1098
GCA	0.0909	0.1347	0.0967	0.1251	0.1069	0.1024	0.0807	0.0893	0.0806	0.0925
GGT	0.1529	0.1287	0.1001	0.0780	0.1000	0.1019	0.0749	0.0703	0.0991	0.0941
GAC	0.1002	0.1190	0.0924	0.0825	0.1079	0.1017	0.0883	0.0988	0.0920	0.1172
AGT	0.0901	0.0923	0.1076	0.1067	0.0623	0.1016	0.1135	0.0848	0.1353	0.1058
ACT	0.0625	0.1032	0.1048	0.1181	0.1075	0.0990	0.1100	0.1117	0.0914	0.0917
TGT	0.0953	0.0681	0.0846	0.0647	0.1306	0.0986	0.1052	0.1041	0.1445	0.1042
TCC	0.0913	0.1085	0.1055	0.1026	0.0721	0.0982	0.1155	0.1156	0.0936	0.0973
ATT	0.0644	0.0773	0.1474	0.1305	0.0726	0.0962	0.1055	0.1195	0.1012	0.0855
AGC	0.0968	0.0882	0.1113	0.1272	0.0695	0.0961	0.1240	0.0798	0.1110	0.0961
CGA	0.0793	0.0941	0.1360	0.1040	0.0915	0.0952	0.0963	0.1604	0.0940	0.0490
TTA	0.0814	0.0779	0.1138	0.1566	0.0771	0.0947	0.1100	0.1247	0.0953	0.0684
TTT	0.0718	0.0627	0.1085	0.1060	0.1320	0.0940	0.1174	0.1269	0.1142	0.0666
AGA	0.1017	0.0746	0.0914	0.0849	0.0726	0.0932	0.1184	0.1052	0.1440	0.1140
TTC	0.0977	0.0985	0.0988	0.0922	0.1100	0.0924	0.0998	0.1234	0.0868	0.1003
TCA	0.0686	0.0929	0.1294	0.1469	0.0821	0.0918	0.1334	0.1230	0.0642	0.0677
ATC	0.0775	0.1030	0.1308	0.1295	0.0688	0.0902	0.0995	0.1114	0.0853	0.1041
GAT	0.0868	0.1076	0.0948	0.0832	0.1185	0.0899	0.1074	0.1074	0.0934	0.1110
GTC	0.1016	0.1273	0.1172	0.1092	0.0807	0.0897	0.0884	0.0872	0.0815	0.1172
ACA	0.0507	0.0747	0.0939	0.1396	0.1055	0.0895	0.0941	0.1500	0.1012	0.1006
GTT	0.0847	0.1226	0.1252	0.1058	0.0735	0.0848	0.0898	0.1027	0.0913	0.1196
ACC	0.1022	0.1172	0.1004	0.1196	0.1002	0.0841	0.1066	0.0968	0.0928	0.0801
ATA	0.0403	0.0371	0.1367	0.2565	0.0712	0.0837	0.1203	0.1191	0.0771	0.0580
AAG	0.1048	0.0627	0.1019	0.1002	0.1036	0.0834	0.1126	0.1096	0.1325	0.0887
ACG	0.0989	0.0774	0.0912	0.1241	0.1291	0.0829	0.0605	0.1126	0.1324	0.0910
GTG	0.1201	0.0919	0.0961	0.1037	0.0799	0.0785	0.0787	0.0828	0.1361	0.1322
TGG	0.1892	0.0787	0.0962	0.1207	0.0926	0.0780	0.0818	0.0758	0.1307	0.0562
CGG	0.1762	0.1163	0.0554	0.0684	0.2499	0.0755	0.0904	0.0478	0.0649	0.0552
CAC	0.1042	0.1071	0.1021	0.1230	0.1115	0.0748	0.1167	0.1013	0.0777	0.0815
CGT	0.1065	0.1108	0.1231	0.0717	0.0697	0.0744	0.0890	0.1041	0.1235	0.1271
ATG	0.1247	0.0895	0.0998	0.1232	0.0842	0.0715	0.0896	0.1157	0.1278	0.0741
GCG	0.1482	0.1007	0.0905	0.1440	0.0933	0.0665	0.0690	0.0764	0.1181	0.0932

Table A.5: Ribosome residence time analysis from the Ingolia expt.

Codon	*Pos*1	*Pos*2	*Pos*3	*Pos*4	*Pos*5	*Pos*6	*Pos*7	*Pos*8
GGG	0.0714	0.1387	0.1067	0.1213	0.1133	0.1693	0.1367	0.1426
GAG	0.0841	0.1365	0.1097	0.1147	0.1320	0.1643	0.1267	0.1320
GGC	0.0662	0.1210	0.1311	0.1236	0.1223	0.1550	0.1382	0.1425
TAT	0.1216	0.1121	0.1191	0.1183	0.1329	0.1543	0.1270	0.1147
GTA	0.0962	0.1185	0.1369	0.1131	0.1025	0.1532	0.1297	0.1500
GAA	0.0954	0.1428	0.1138	0.1292	0.1280	0.1507	0.1193	0.1208
TAC	0.1005	0.1216	0.1297	0.1249	0.1124	0.1497	0.1355	0.1257
GGA	0.0858	0.1327	0.1156	0.1240	0.1392	0.1493	0.1319	0.1213
TTG	0.0733	0.1258	0.1299	0.1271	0.1327	0.1475	0.1387	0.1251
AGG	0.1731	0.1248	0.1075	0.1035	0.0827	0.1449	0.1300	0.1334
GTG	0.0840	0.1302	0.1257	0.1082	0.1183	0.1415	0.1374	0.1546
CCA	0.1315	0.0948	0.1118	0.1297	0.1524	0.1394	0.1254	0.1150
GGT	0.0824	0.1307	0.1137	0.1192	0.1459	0.1389	0.1312	0.1380
AAA	0.1625	0.1218	0.1027	0.1208	0.1322	0.1381	0.1166	0.1054
AGC	0.1735	0.1158	0.1164	0.1151	0.0871	0.1348	0.1254	0.1319
TCG	0.0901	0.1305	0.1117	0.1232	0.1218	0.1341	0.1478	0.1407
GTT	0.0905	0.1306	0.1551	0.1180	0.1044	0.1330	0.1319	0.1366
CCG	0.1217	0.1164	0.0979	0.1174	0.1337	0.1327	0.1435	0.1368
TTA	0.0957	0.1208	0.1312	0.1401	0.1318	0.1318	0.1303	0.1181
ATT	0.1753	0.1129	0.1275	0.1125	0.1185	0.1293	0.1139	0.1101
GAC	0.0700	0.1376	0.1370	0.1333	0.1380	0.1293	0.1226	0.1321
ATA	0.1765	0.1121	0.1343	0.1228	0.0824	0.1291	0.1304	0.1124
TCA	0.1027	0.1156	0.1282	0.1413	0.1179	0.1289	0.1361	0.1293
TCC	0.0817	0.1278	0.1299	0.1340	0.1243	0.1288	0.1396	0.1339
GTC	0.0756	0.1345	0.1501	0.1180	0.0937	0.1286	0.1396	0.1599
GAT	0.0877	0.1265	0.1214	0.1282	0.1559	0.1281	0.1221	0.1302
GCA	0.1115	0.1304	0.1393	0.1296	0.0957	0.1280	0.1266	0.1390
TTC	0.0771	0.1391	0.1446	0.1331	0.1145	0.1278	0.1349	0.1290
ATC	0.1666	0.1184	0.1277	0.1122	0.1052	0.1259	0.1206	0.1233
AAG	0.1816	0.1097	0.0999	0.1176	0.1325	0.1242	0.1130	0.1214
CTG	0.1176	0.1096	0.1358	0.1344	0.1108	0.1240	0.1345	0.1333
TCT	0.0918	0.1284	0.1345	0.1300	0.1405	0.1237	0.1314	0.1199
TGC	0.0732	0.1233	0.1337	0.1227	0.1248	0.1237	0.1588	0.1399
GCC	0.0714	0.1471	0.1474	0.1323	0.0979	0.1231	0.1314	0.1492
CTC	0.1081	0.1029	0.1555	0.1472	0.0915	0.1230	0.1408	0.1309
TTT	0.0950	0.1362	0.1415	0.1298	0.1240	0.1226	0.1340	0.1169
CTA	0.1205	0.1073	0.1468	0.1397	0.1139	0.1219	0.1336	0.1163
AGT	0.1673	0.1153	0.1091	0.1182	0.1039	0.1218	0.1394	0.1250
AGA	0.1947	0.1243	0.1048	0.1080	0.1086	0.1216	0.1158	0.1224
CGC	0.1277	0.1173	0.1706	0.1204	0.0906	0.1182	0.1209	0.1342
CCC	0.0993	0.1119	0.1327	0.1399	0.1479	0.1147	0.1268	0.1268
GCT	0.0930	0.1463	0.1446	0.1276	0.1123	0.1139	0.1234	0.1390
TGG	0.0846	0.1418	0.1283	0.1216	0.1009	0.1136	0.1579	0.1512
ACG	0.1669	0.1320	0.1099	0.1164	0.1276	0.1127	0.1073	0.1272
ATG	0.1989	0.1196	0.1136	0.1191	0.1015	0.1124	0.1129	0.1220
AAC	0.1697	0.1230	0.1177	0.1204	0.1395	0.1101	0.1070	0.1125
CCT	0.1141	0.1062	0.1195	0.1307	0.1761	0.1099	0.1281	0.1154
AAT	0.1698	0.1135	0.1098	0.1185	0.1587	0.1095	0.1115	0.1087
CTT	0.1111	0.1081	0.1583	0.1424	0.1227	0.1079	0.1271	0.1225
GCG	0.0939	0.1616	0.1317	0.1379	0.0755	0.1041	0.1293	0.1662
TGT	0.0949	0.1290	0.1226	0.1134	0.1566	0.1024	0.1471	0.1341
ACC	0.1565	0.1313	0.1293	0.1308	0.1080	0.1016	0.1151	0.1274
CAG	0.1558	0.0971	0.1144	0.1408	0.1352	0.0976	0.1263	0.1329
CGT	0.1361	0.1300	0.1374	0.1211	0.1064	0.0970	0.1374	0.1346
ACA	0.1817	0.1202	0.1255	0.1281	0.1163	0.0942	0.1139	0.1201
ACT	0.1793	0.1366	0.1238	0.1266	0.1151	0.0911	0.1124	0.1150
CGG	0.0959	0.0981	0.1347	0.1087	0.2116	0.0894	0.1183	0.1433
CGA	0.1394	0.1467	0.1323	0.1335	0.0687	0.0865	0.1438	0.1491
CAA	0.2000	0.0999	0.1211	0.1363	0.1206	0.0806	0.1188	0.1227
CAT	0.1566	0.1082	0.1306	0.1381	0.1405	0.0710	0.1271	0.1279
CAC	0.1603	0.1124	0.1280	0.1453	0.1221	0.0703	0.1231	0.1385

Table A.6: Ribosome residence time for short footprints (aniso2 dataset).

Codon	*Pos*1	*Pos*2	*Pos*3	*Pos*4	*Pos*5	*Pos*6	*Pos*7	*#Windows*
CTT	0.0972	0.1200	0.1257	0.1331	0.2226	0.1477	0.1537	5092
GCC	0.0732	0.1898	0.1928	0.2062	0.1064	0.1097	0.1221	6575
GGA	0.0677	0.2183	0.2411	0.1452	0.1007	0.0693	0.1577	3628
GTC	0.0907	0.1818	0.1492	0.1284	0.1788	0.1245	0.1466	5416
TGC	0.1721	0.1689	0.1769	0.1910	0.0890	0.0645	0.1376	1876
AGT	0.0842	0.1273	0.1315	0.1728	0.2236	0.1422	0.1183	6481
TGT	0.1606	0.1460	0.1451	0.1841	0.1148	0.1160	0.1335	3045
TCA	0.1973	0.1192	0.1193	0.1445	0.2029	0.0901	0.1268	8619
CGA	0.1156	0.0933	0.1457	0.0741	0.1245	0.3126	0.1342	1167
ATT	0.1593	0.1149	0.1165	0.1503	0.1453	0.1407	0.1731	11355
TAT	0.2026	0.1296	0.1253	0.1676	0.1014	0.1571	0.1165	7220
ATC	0.1938	0.1339	0.1159	0.1544	0.1292	0.1275	0.1453	8547
AAC	0.1926	0.1783	0.1491	0.1012	0.1132	0.1273	0.1382	9926
AGC	0.0996	0.1545	0.1537	0.1823	0.1858	0.1050	0.1191	4603
TAC	0.2353	0.1486	0.1342	0.1813	0.1081	0.0841	0.1085	7017
AAT	0.1572	0.1457	0.1396	0.0884	0.1148	0.1999	0.1543	10919
ACT	0.1600	0.1322	0.1281	0.1578	0.1036	0.1485	0.1699	9807
TCG	0.1746	0.1099	0.1168	0.1275	0.1946	0.1341	0.1424	4909
ACA	0.2058	0.1376	0.1364	0.1541	0.1030	0.1346	0.1285	9038
GAC	0.0673	0.1986	0.1786	0.1102	0.1558	0.1560	0.1336	7804
CAA	0.1953	0.1600	0.1660	0.1622	0.0902	0.1249	0.1014	9871
CCG	0.1247	0.1472	0.1246	0.1456	0.0576	0.2435	0.1567	2857
CTG	0.1155	0.1286	0.1258	0.1277	0.1966	0.1620	0.1437	4791
GGT	0.0764	0.2207	0.2405	0.1592	0.1229	0.0685	0.1117	6212
GCA	0.0812	0.1788	0.1922	0.2199	0.1053	0.1075	0.1150	8004
AAG	0.1769	0.1158	0.1387	0.0752	0.1011	0.2293	0.1629	10826
GTG	0.0799	0.1522	0.1280	0.1332	0.2521	0.1246	0.1299	4371
TCC	0.1756	0.1290	0.1185	0.1421	0.1824	0.0969	0.1556	7729
TTT	0.1772	0.1129	0.1083	0.1611	0.1329	0.1120	0.1955	9261
AGG	0.1207	0.1088	0.1488	0.0965	0.0921	0.2553	0.1777	3592
CAC	0.1875	0.1633	0.1736	0.1574	0.0970	0.0997	0.1216	4053
GTT	0.0731	0.1477	0.1547	0.1412	0.1857	0.1269	0.1707	7972
CGT	0.1118	0.1190	0.1737	0.0931	0.1276	0.2415	0.1333	2565
CGG	0.1204	0.1301	0.1672	0.0832	0.0654	0.3132	0.1204	575
AGA	0.1235	0.1158	0.1451	0.0844	0.1025	0.2103	0.2184	7851
CAT	0.1468	0.1424	0.1625	0.1625	0.0980	0.1800	0.1078	5742
ATA	0.1678	0.1105	0.1028	0.1465	0.2125	0.1347	0.1252	8016
GGG	0.0835	0.2220	0.2574	0.1479	0.0952	0.0695	0.1245	1924
CCC	0.1285	0.1406	0.1253	0.1771	0.1126	0.1520	0.1640	3769
ACC	0.1905	0.1463	0.1371	0.1531	0.0989	0.1374	0.1367	7013
GAG	0.0684	0.1782	0.1775	0.1309	0.0665	0.2363	0.1421	6257
TTA	0.1822	0.1062	0.0918	0.1043	0.2935	0.0945	0.1275	10995
CCA	0.1656	0.1611	0.1327	0.2032	0.1089	0.1027	0.1257	9050
CTA	0.1252	0.1323	0.1222	0.1346	0.2791	0.0991	0.1076	6658
GAT	0.0468	0.1606	0.1602	0.0968	0.1397	0.2451	0.1507	10457
TCT	0.1472	0.1138	0.1159	0.1467	0.1958	0.1003	0.1804	10345
TGG	0.2147	0.1174	0.1255	0.1184	0.1021	0.1829	0.1390	3245
TTC	0.1984	0.1321	0.1061	0.1751	0.1463	0.0769	0.1652	8394
CTC	0.1062	0.1227	0.1203	0.1419	0.2906	0.1045	0.1137	2779
CGC	0.1422	0.1343	0.1784	0.1030	0.0997	0.1787	0.1637	1010
TTG	0.1693	0.0963	0.0910	0.1132	0.2514	0.1103	0.1686	10815
GCG	0.0826	0.1545	0.1780	0.1977	0.0724	0.1914	0.1234	3152
GGC	0.0707	0.2499	0.2557	0.1523	0.0990	0.0484	0.1240	3463
GCT	0.0663	0.1556	0.1718	0.2016	0.1159	0.1310	0.1579	8709
CAG	0.1386	0.1618	0.1681	0.1841	0.0690	0.1532	0.1253	5547
GAA	0.0764	0.1947	0.1842	0.1325	0.0886	0.1861	0.1375	11360
CCT	0.1156	0.1543	0.1365	0.1761	0.1177	0.1272	0.1727	6913
ACG	0.1911	0.1230	0.1275	0.1667	0.0876	0.1516	0.1525	4660
ATG	0.2038	0.0958	0.1072	0.1854	0.1341	0.0971	0.1767	7806
AAA	0.1813	0.1100	0.1298	0.0724	0.1302	0.2391	0.1371	11988
GTA	0.0799	0.1542	0.1377	0.1320	0.2547	0.1084	0.1330	5165

Table A.7: Ribosome residence time for short footprints (aniso1B dataset).

Codon	*Pos*1	*Pos*2	*Pos*3	*Pos*4	*Pos*5	*Pos*6	*Pos*7	*#Windows*
GCC	0.1365	0.1570	0.1787	0.2132	0.0984	0.0936	0.1226	11754
CTT	0.1493	0.1052	0.1222	0.1492	0.2156	0.1224	0.1361	9758
GGA	0.1885	0.1808	0.2317	0.1023	0.1123	0.0383	0.1461	7670
GTC	0.1395	0.1331	0.1498	0.1280	0.2117	0.0978	0.1401	10156
TGC	0.1751	0.1600	0.1539	0.1845	0.1083	0.0931	0.1252	3668
AGT	0.1344	0.1382	0.1247	0.1694	0.1428	0.1377	0.1527	11094
TGT	0.1577	0.1452	0.1443	0.1772	0.1235	0.1010	0.1511	5941
TCA	0.1374	0.1383	0.1321	0.1757	0.1634	0.1232	0.1298	13594
CGA	0.1190	0.0732	0.1008	0.0465	0.1411	0.3723	0.1472	2603
ATT	0.1309	0.1335	0.1205	0.1672	0.1841	0.1039	0.1598	18779
TAT	0.1553	0.1312	0.1122	0.1774	0.1251	0.1646	0.1341	12461
ATC	0.1075	0.1429	0.1201	0.1717	0.1875	0.1176	0.1526	13882
AAC	0.1376	0.1809	0.1458	0.0828	0.1192	0.1475	0.1862	16192
AGC	0.1419	0.1376	0.1391	0.1952	0.1420	0.1234	0.1206	8373
TAC	0.1268	0.1444	0.1305	0.1985	0.1306	0.1294	0.1399	11335
AAT	0.1319	0.1714	0.1378	0.0826	0.1206	0.1765	0.1791	18713
TCG	0.1206	0.1277	0.1265	0.1713	0.1498	0.1784	0.1259	8317
ACT	0.1390	0.1329	0.1261	0.1722	0.1439	0.1175	0.1684	15849
ACA	0.1491	0.1471	0.1303	0.1628	0.1368	0.1341	0.1398	13864
GAC	0.1190	0.1733	0.1799	0.0782	0.1225	0.1750	0.1521	14939
CAA	0.1687	0.1422	0.1463	0.1580	0.1000	0.1589	0.1259	17109
CCG	0.1348	0.1335	0.1079	0.1646	0.0519	0.2737	0.1336	5598
CTG	0.1580	0.1196	0.1138	0.1234	0.1754	0.1836	0.1262	9475
GCA	0.1538	0.1428	0.1704	0.2011	0.1056	0.1064	0.1199	14211
GGT	0.1948	0.1751	0.2359	0.1108	0.1246	0.0293	0.1295	12960
GTG	0.1578	0.1411	0.1496	0.1203	0.2207	0.1059	0.1047	8208
AAG	0.1448	0.1221	0.1337	0.0559	0.1163	0.2495	0.1777	18417
TCC	0.1116	0.1396	0.1333	0.1971	0.1582	0.1180	0.1422	12710
TTT	0.1392	0.1315	0.1205	0.1929	0.1753	0.0858	0.1547	16150
CAC	0.1548	0.1276	0.1502	0.1585	0.1058	0.1592	0.1440	6960
AGG	0.1498	0.1256	0.1539	0.0628	0.1025	0.2323	0.1731	7245
GTT	0.1503	0.1302	0.1562	0.1256	0.2115	0.0853	0.1409	15355
CGT	0.1398	0.0961	0.1428	0.0642	0.1321	0.2341	0.1910	5496
CGG	0.1310	0.0964	0.1240	0.0602	0.0809	0.3498	0.1576	1466
ATA	0.1205	0.1329	0.1157	0.1715	0.2031	0.1302	0.1262	12373
AGA	0.1578	0.1236	0.1384	0.0567	0.1029	0.2206	0.1999	13981
CAT	0.1477	0.1194	0.1373	0.1550	0.1087	0.1881	0.1439	10656
ACC	0.1160	0.1498	0.1343	0.1888	0.1384	0.1183	0.1544	11375
CCC	0.1137	0.1387	0.1245	0.2363	0.0754	0.1783	0.1331	7395
GGG	0.1876	0.1956	0.2644	0.1089	0.1013	0.0301	0.1121	4512
TTA	0.1329	0.1227	0.1128	0.1307	0.2752	0.1072	0.1184	17302
GAG	0.1435	0.1575	0.1582	0.0981	0.0831	0.2123	0.1473	13090
CCA	0.1505	0.1460	0.1267	0.2341	0.0857	0.1301	0.1268	14918
CTA	0.1491	0.1146	0.1216	0.1492	0.2365	0.1079	0.1212	10891
GAT	0.1322	0.1575	0.1550	0.0736	0.1060	0.2125	0.1631	20267
TCT	0.1317	0.1308	0.1299	0.1708	0.1614	0.1149	0.1604	17120
TGG	0.1836	0.1230	0.1192	0.0993	0.1059	0.1998	0.1691	6804
TTC	0.1232	0.1371	0.1224	0.2035	0.1918	0.0778	0.1443	13934
CTC	0.1304	0.1022	0.1207	0.1578	0.2731	0.1065	0.1094	5359
CGC	0.1492	0.0987	0.1543	0.0758	0.1229	0.2457	0.1533	2244
TTG	0.1234	0.1277	0.1152	0.1319	0.2289	0.1296	0.1433	18790
GCG	0.1446	0.1381	0.1594	0.1680	0.0826	0.1878	0.1194	6486
GGC	0.1864	0.1826	0.2519	0.1079	0.1150	0.0327	0.1234	8181
GCT	0.1485	0.1405	0.1647	0.1985	0.1071	0.0981	0.1427	16615
GAA	0.1637	0.1671	0.1513	0.1000	0.0873	0.1818	0.1488	21473
CAG	0.1581	0.1487	0.1484	0.1433	0.0855	0.1839	0.1321	10246
CCT	0.1515	0.1381	0.1226	0.2171	0.0856	0.1335	0.1516	12270
ACG	0.1241	0.1420	0.1318	0.1710	0.1349	0.1545	0.1418	7604
ATG	0.1339	0.1283	0.1291	0.2252	0.1450	0.0904	0.1481	12791
AAA	0.1244	0.1213	0.1327	0.0580	0.1385	0.2593	0.1659	20049
GTA	0.1365	0.1346	0.1468	0.1227	0.2426	0.1001	0.1168	9512

Table A.8: Ribosome residence time for short footprints (aniso1A dataset).

Codon	*Pos*1	*Pos*2	*Pos*3	*Pos*4	*Pos*5	*Pos*6	*Pos*7	*#Windows*
GCC	0.1447	0.1599	0.1779	0.2131	0.0963	0.0893	0.1189	10597
CTT	0.1468	0.1071	0.1250	0.1461	0.2199	0.1203	0.1348	8311
GGA	0.1837	0.1783	0.2375	0.0994	0.1068	0.0374	0.1569	6184
GTC	0.1421	0.1408	0.1498	0.1265	0.2098	0.0967	0.1343	8803
TGC	0.1781	0.1528	0.1574	0.1804	0.1101	0.0827	0.1385	3132
AGT	0.1343	0.1351	0.1243	0.1752	0.1488	0.1323	0.1499	9303
TGT	0.1600	0.1489	0.1389	0.1790	0.1269	0.1027	0.1436	4913
TCA	0.1458	0.1350	0.1296	0.1722	0.1633	0.1277	0.1264	11629
CGA	0.1106	0.0741	0.1003	0.0457	0.1401	0.3966	0.1327	2165
ATT	0.1304	0.1304	0.1190	0.1645	0.1899	0.1045	0.1614	15718
TAT	0.1529	0.1312	0.1102	0.1744	0.1287	0.1704	0.1323	10387
ATC	0.1086	0.1464	0.1205	0.1717	0.1825	0.1162	0.1541	11892
AAC	0.1381	0.1823	0.1484	0.0793	0.1140	0.1460	0.1918	13726
AGC	0.1472	0.1387	0.1439	0.1830	0.1335	0.1144	0.1392	7233
TAC	0.1306	0.1439	0.1296	0.1982	0.1343	0.1315	0.1320	9708
AAT	0.1321	0.1690	0.1375	0.0797	0.1220	0.1811	0.1785	15408
TCG	0.1230	0.1301	0.1289	0.1646	0.1520	0.1748	0.1266	7154
ACT	0.1445	0.1330	0.1253	0.1716	0.1428	0.1151	0.1678	13644
ACA	0.1532	0.1519	0.1276	0.1569	0.1367	0.1352	0.1386	11859
GAC	0.1229	0.1729	0.1731	0.0805	0.1215	0.1729	0.1563	12710
CAA	0.1684	0.1465	0.1479	0.1572	0.1033	0.1603	0.1165	14515
CCG	0.1340	0.1325	0.1106	0.1676	0.0490	0.2713	0.1349	4884
CTG	0.1593	0.1231	0.1124	0.1212	0.1696	0.1874	0.1270	8128
GGT	0.1871	0.1726	0.2397	0.1090	0.1256	0.0289	0.1371	10948
GCA	0.1564	0.1488	0.1723	0.1994	0.1031	0.1041	0.1159	12368
GTG	0.1502	0.1405	0.1430	0.1214	0.2283	0.1042	0.1124	7021
AAG	0.1394	0.1168	0.1348	0.0514	0.1148	0.2549	0.1880	15454
TCC	0.1197	0.1433	0.1347	0.1926	0.1553	0.1152	0.1392	11174
TTT	0.1403	0.1322	0.1181	0.1977	0.1756	0.0859	0.1503	13662
AGG	0.1519	0.1218	0.1551	0.0584	0.0979	0.2275	0.1873	6054
CAC	0.1690	0.1350	0.1458	0.1561	0.1068	0.1507	0.1367	6119
GTT	0.1449	0.1310	0.1572	0.1237	0.2164	0.0860	0.1408	13103
CGT	0.1369	0.1025	0.1409	0.0645	0.1349	0.2357	0.1845	4630
CGG	0.1366	0.0906	0.1419	0.0477	0.0727	0.3345	0.1760	1200
ATA	0.1203	0.1309	0.1129	0.1682	0.2058	0.1374	0.1246	9966
CAT	0.1495	0.1244	0.1442	0.1522	0.1045	0.1896	0.1356	8996
AGA	0.1541	0.1216	0.1355	0.0530	0.1035	0.2299	0.2024	11466
ACC	0.1198	0.1497	0.1318	0.1856	0.1405	0.1188	0.1538	10126
CCC	0.1218	0.1417	0.1168	0.2362	0.0760	0.1723	0.1352	6465
GGG	0.1805	0.1941	0.2777	0.1076	0.0884	0.0273	0.1244	3730
TTA	0.1325	0.1244	0.1095	0.1328	0.2823	0.1064	0.1120	14669
GAG	0.1448	0.1540	0.1609	0.0944	0.0787	0.2131	0.1541	10696
CCA	0.1546	0.1490	0.1255	0.2336	0.0843	0.1296	0.1234	13135
GAT	0.1233	0.1509	0.1557	0.0729	0.1074	0.2183	0.1715	16623
CTA	0.1525	0.1198	0.1232	0.1468	0.2407	0.1091	0.1079	9308
TCT	0.1344	0.1315	0.1312	0.1736	0.1588	0.1133	0.1573	14629
TGG	0.1821	0.1207	0.1160	0.0983	0.0966	0.1979	0.1885	5662
TTC	0.1284	0.1387	0.1263	0.2067	0.1871	0.0752	0.1376	12087
CTC	0.1339	0.1019	0.1179	0.1624	0.2798	0.0988	0.1053	4636
CGC	0.1645	0.0950	0.1583	0.0740	0.1175	0.2409	0.1498	1894
TTG	0.1196	0.1230	0.1118	0.1286	0.2330	0.1317	0.1522	15957
GCG	0.1461	0.1349	0.1579	0.1774	0.0806	0.1829	0.1201	5492
GGC	0.1828	0.1765	0.2628	0.1061	0.1119	0.0259	0.1340	7065
GCT	0.1487	0.1431	0.1673	0.1997	0.1050	0.0966	0.1396	14450
CAG	0.1611	0.1570	0.1424	0.1475	0.0818	0.1821	0.1280	9047
GAA	0.1575	0.1639	0.1532	0.0982	0.0891	0.1855	0.1525	17674
CCT	0.1577	0.1467	0.1223	0.2170	0.0818	0.1301	0.1445	10942
ACG	0.1228	0.1442	0.1300	0.1709	0.1330	0.1510	0.1479	6576
AAA	0.1179	0.1140	0.1305	0.0560	0.1405	0.2763	0.1648	16232
ATG	0.1329	0.1226	0.1304	0.2234	0.1453	0.0872	0.1581	10629
GTA	0.1310	0.1341	0.1468	0.1211	0.2506	0.1015	0.1150	7884

Table A.9: Weighted mean RRT of codon pair groups, considering scoring methods: classic, gap, signal.

Scoring method - classic:

Position	high	p-value (high)	low	p-value (low)
1	0.098	0.017	0.103	0.153
2	0.103	0.001	0.105	0.001
3	0.103	0.183	0.103	0.234
4	0.101	0.297	0.100	0.511
5	0.096	0.104	0.098	0.588
6	0.095	0.130	0.092	0.001
7	0.098	0.580	0.100	0.001
8	0.099	0.012	0.104	0.005
9	0.102	0.917	0.097	0.001
10	0.104	0.002	0.098	0.279

Scoring method - gap:

Position	high	p-value (high)	low	p-value (low)
1	0.101	0.516	0.101	0.381
2	0.104	0.244	0.107	0.002
3	0.103	0.445	0.102	0.869
4	0.098	0.353	0.099	0.902
5	0.096	0.437	0.096	0.423
6	0.093	0.284	0.089	0.001
7	0.097	0.847	0.101	0.001
8	0.101	0.887	0.103	0.235
9	0.102	0.750	0.098	0.001
10	0.106	0.140	0.104	0.579

Scoring method - signal:

Position	high	p-value (high)	low	p-value (low)
1	0.101	0.893	0.104	0.41
2	0.104	0.003	0.106	0.001
3	0.104	0.267	0.104	0.576
4	0.102	0.276	0.101	0.595
5	0.096	0.638	0.097	0.698
6	0.089	0.001	0.095	0.515
7	0.097	0.199	0.098	0.482
8	0.102	0.131	0.100	0.583
9	0.101	0.449	0.100	0.125
10	0.103	0.100	0.097	0.019

A.3 Effect of RNA secondary structure on ribosome profile data

Correlation scores of *Saccharomyces cerevisiae* RNA secondary structure energy vs control data-sets (in silico random data and wildtype mRNA-seq data) at different offsets of energy window from the read window have been listed in table A.10 and A.11.

Table A.10: Correlation table for *Saccharomyces cerevisiae* RNA secondary structure energy vs randomly generated read data-set.

Offset	Frequency	Spearman	P-value	Pearson	P-value	K. tau	P-value
-15	437769	-0.0003	0.854	-0.0009	0.567	-0.0002	0.845
-14	438271	-0.0004	0.799	-0.0002	0.878	-0.0003	0.785
-13	438773	-0.0005	0.730	0.0006	0.703	-0.0004	0.709
-12	439275	0.0004	0.788	0.0011	0.446	0.0003	0.769
-11	439777	0.0007	0.655	0.0011	0.448	0.0005	0.630
-10	440279	0.0009	0.551	0.0010	0.487	0.0006	0.519
-9	440781	0.0012	0.439	0.0009	0.572	0.0008	0.401
-8	441283	0.0014	0.354	0.0009	0.547	0.0010	0.315
-7	441785	0.0019	0.199	0.0012	0.406	0.0014	0.165
-6	442287	0.0019	0.216	0.0014	0.349	0.0013	0.181
-5	442789	0.0019	0.204	0.0015	0.318	0.0014	0.169
-4	443291	0.0011	0.468	0.0010	0.499	0.0008	0.431
-3	443793	0.0017	0.257	0.0013	0.394	0.0012	0.219
-2	444295	0.0027	0.074	0.0018	0.242	0.0019	0.053
-1	444797	0.0034	0.023	0.0023	0.121	0.0025	0.014
0	445299	0.0033	0.023	0.0024	0.116	0.0024	0.018
1	444797	0.0029	0.050	0.0021	0.169	0.0021	0.034
2	444295	0.0029	0.051	0.0021	0.162	0.0021	0.034
3	443793	0.0017	0.270	0.0019	0.212	0.0012	0.231
4	443291	0.0010	0.492	0.0013	0.371	0.0007	0.455
5	442789	0.0005	0.745	0.0007	0.663	0.0004	0.721
6	442287	0.0009	0.559	0.0005	0.721	0.0006	0.525
7	441785	0.0007	0.638	0.0004	0.812	0.0005	0.608
8	441283	0.0005	0.754	0.0002	0.878	0.0003	0.731
9	440781	0.0004	0.796	0.0000	0.998	0.0003	0.774
10	440279	0.0002	0.884	0.0001	0.946	0.0002	0.869
11	439777	0.0004	0.779	0.0006	0.694	0.0003	0.755
12	439275	0.0005	0.755	0.0009	0.560	0.0003	0.729
13	438773	0.0005	0.755	0.0010	0.493	0.0003	0.728
14	438271	0.0002	0.908	0.0007	0.645	0.0001	0.893
15	437769	0.0002	0.901	0.0008	0.618	0.0001	0.886

Table A.11: Correlation table for *Saccharomyces cerevisiae* RNA secondary structure energy vs wildtype mRNA-seq data-set.

Offset	Frequency	Spearman	P-value	Pearson	P-value	K. tau	P-value
-15	195631	-0.017	2.15E-13	-0.005	0.0386	-0.011	3.98E-14
-14	196131	-0.018	3.50E-16	-0.006	0.0094	-0.013	4.29E-17
-13	196631	-0.019	8.14E-17	-0.007	0.0028	-0.013	9.13E-18
-12	197131	-0.018	2.17E-16	-0.007	0.0010	-0.013	2.58E-17
-11	197631	-0.018	3.04E-16	-0.008	0.0002	-0.013	3.68E-17
-10	198131	-0.019	1.50E-16	-0.010	3.12E-06	-0.013	1.68E-17
-9	198631	-0.022	3.12E-22	-0.014	2.95E-10	-0.015	1.58E-23
-8	199131	-0.026	2.82E-31	-0.018	1.51E-15	-0.018	3.86E-33
-7	199631	-0.029	1.54E-37	-0.020	2.78E-19	-0.020	8.49E-40
-6	200131	-0.029	1.81E-37	-0.019	1.24E-17	-0.020	1.06E-39
-5	200631	-0.028	6.90E-35	-0.017	3.07E-14	-0.019	5.98E-37
-4	201131	-0.026	1.94E-32	-0.016	1.44E-12	-0.018	2.47E-34
-3	201631	-0.027	2.37E-34	-0.017	6.81E-14	-0.019	2.45E-36
-2	202131	-0.027	2.09E-33	-0.018	2.40E-15	-0.018	2.38E-35
-1	202631	-0.025	2.78E-30	-0.017	4.74E-15	-0.017	4.38E-32
0	203131	-0.026	2.77E-32	-0.017	2.08E-14	-0.018	2.91E-34
1	202631	-0.029	2.13E-38	-0.018	1.36E-15	-0.020	9.63E-41
2	202131	-0.032	3.38E-47	-0.020	1.65E-19	-0.022	4.51E-50
3	201631	-0.033	2.14E-49	-0.021	4.91E-21	-0.023	2.19E-52
4	201131	-0.032	1.67E-46	-0.020	3.09E-19	-0.022	2.54E-49
5	200631	-0.030	3.16E-41	-0.017	3.55E-14	-0.021	9.68E-44
6	200131	-0.028	8.17E-36	-0.014	6.33E-10	-0.019	5.78E-38
7	199631	-0.027	2.30E-33	-0.011	5.03E-07	-0.019	2.54E-35
8	199131	-0.026	5.42E-32	-0.010	1.04E-05	-0.018	7.95E-34
9	198631	-0.026	1.82E-31	-0.009	5.89E-05	-0.018	2.71E-33
10	198131	-0.026	5.25E-30	-0.008	0.0003	-0.018	9.20E-32
11	197631	-0.024	1.40E-27	-0.007	0.0029	-0.017	3.37E-29
12	197131	-0.023	2.68E-24	-0.005	0.0163	-0.016	9.90E-26
13	196631	-0.022	6.66E-23	-0.004	0.0530	-0.015	3.06E-24
14	196131	-0.021	2.22E-21	-0.003	0.1464	-0.015	1.27E-22
15	195631	-0.021	2.30E-20	-0.002	0.3090	-0.014	1.55E-21

Appendix B

Supplementary figures

B.1 Ribosomal pauses and other statistics per gene

Fig. B.1 is showing ribosomal pauses and corresponding potential factors at each codon position of *YAL038W*. The first plot is showing the raw footprint count at each codon position by blue dotted line and a 10 codon-window moving average of it in yellow. We detect the peak regions from the smoothed footprint data. Regions having 1.7 times or more footprints than the average and having at least 200 footprints are selected as peaks (black circles on the smoothed curve). The next plot is showing the folding energy (summed over 3 nucleotides) at each codon position of *YAL038W*. The third image is showing the presence of amino acids *Met*, *Pro* and *ProPro* dipeptide. In the fourth graph, 10 codon-window moving average of the relative adaptiveness of the codons in the gene are shown. Similar plot is shown in fig B.2 for gene *YKL152C*. We see the peaks or read pile-ups in *YAL038W* are comparatively larger than those in *YKL152C*. Note the RNA folding energies are also lower in *YAL038W*. Also, the density of *Met* and *Pro* are much higher in *YAL038W* compared to *YKL152C*. Note for *YKL152C*, folded secondary structure energies near the detected peaks are lower compared to other regions.

B.2 RNA secondary structure effect on ribosome profile data

Fig. B.3 is showing the correlation scores of ribosome footprint reads with inverted PARS scores of the *Saccharomyces cerevisiae* mRNA sequences at different offsets. Left figure is showing correlation statistics for offsets in the

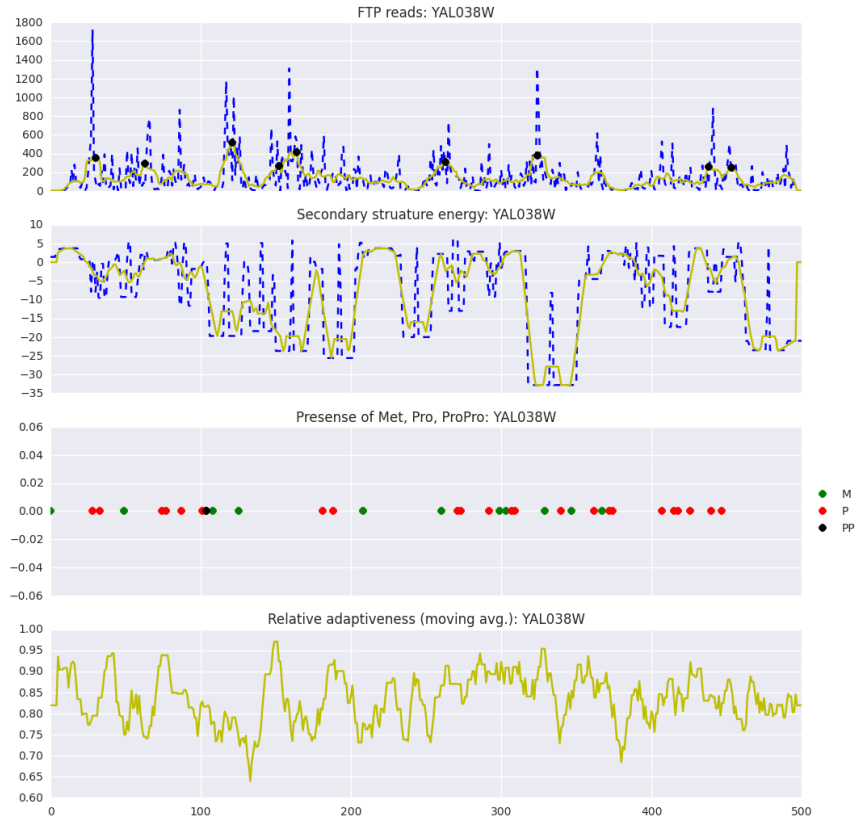

Figure B.1: Ribosomal pauses and other statistics for *YAL038W*. Figure-1: raw read count along the gene is shown in blues dotted line and the 10-codon window smoothed line is shown in yellow. Figure-2: secondary structure energy per codon (raw values in blue dotted line and the smoothed curve in yellow). Figure-3: positions of amino acids *Met*, *Pro* and *ProPro* dipeptide are shown. Figure-4: 10-codon window smoothed line for relative adaptiveness of the RNA sequence is shown.

range: $-20 <= x <= 20$. Right figure is showing statistics for the extended offset range ($-100 <= x <= 100$). We see a significant two dip pattern around the A-site of a translating ribosome. Fig. B.4 is showing the plots of the correlation statistics for control data-set. Left figure is showing the statistics for randomly generated read data-set, we found no significant correlation of the inverted PARS scores with the in silico random read data-set. Right figure is showing the correlation statistics for wildtype mRNA-seq data, it shows significant two dip pattern similar to the real ribosome profile data-set, though less significant compared to the real footprint data.

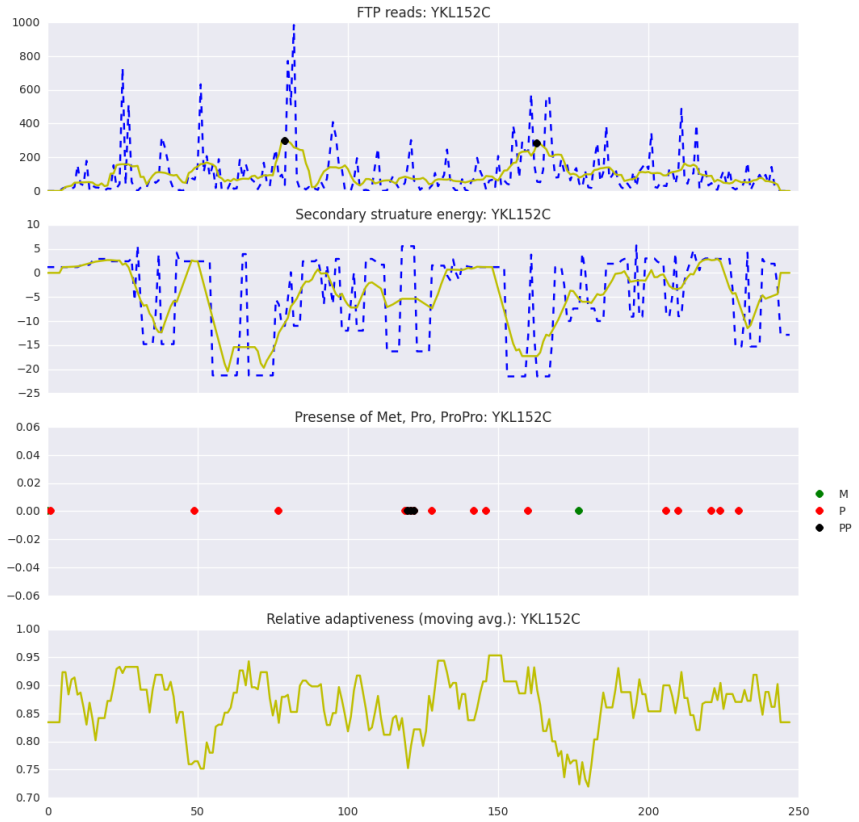

Figure B.2: Ribosomal pauses and other statistics for *YKL152C*. Figure-1: raw read count along the gene is shown in blues dotted line and the 10-codon window smoothed line is shown in yellow. Figure-2: secondary structure energy per codon (raw values in blue dotted line and the smoothed curve in yellow). Figure-3: positions of amino acids *Met*, *Pro* and *ProPro* dipeptide are shown. Figure-4: 10-codon window smoothed line for relative adaptiveness of the RNA sequence is shown.

B.3 Codon-pair bias analysis

Fig. B.5, B.6 and B.7 are showing the patterns of the codon usage statistics for both equal weight and weighted analysis. For equal weight analysis, we took the mean RRT of the codons in the *high* or *low* group (see chapter 5 for codon usage RRT calculation). Here, both groups will have same codons equal number of times, so the mean should also be same. For weighted analysis, we calculated the weighted average (by the frequency of corresponding codon pair) of the codon usage RRT of each codon in a group. The mean for weighted analysis may vary based on the frequency of usage of individual codon pair. However, we found no significant difference in the two groups while considering codon-usage effect.

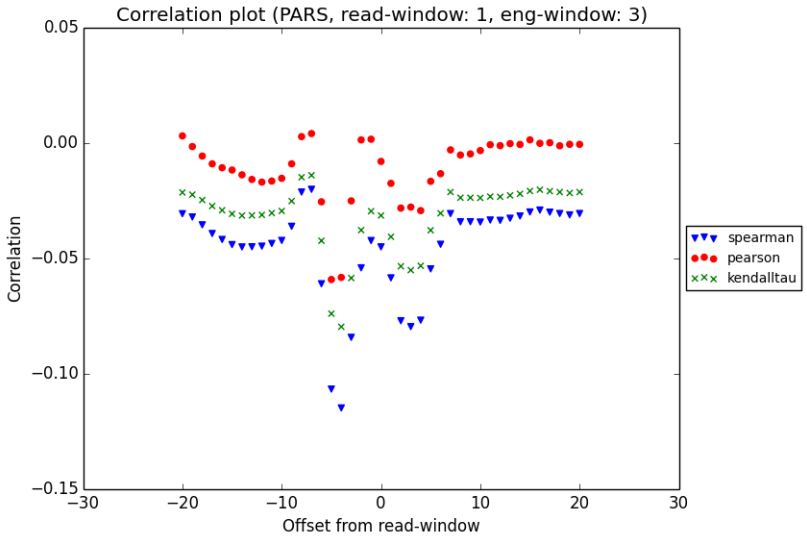

Figure B.3: *Pearson*, *Spearman* and *Kendall Tau* correlation plot for the inverted PARS scores of the *Saccharomyces cerevisiae* genes vs ribosome footprint reads (read-window size: 1 and PARS score window size: 3). Reads are normalized by the average reads per codon. Correlation plots are shown for offsets in the range, $-20 <= x <= 20$, significant two dip pattern is noticed in the correlation values at different offsets.

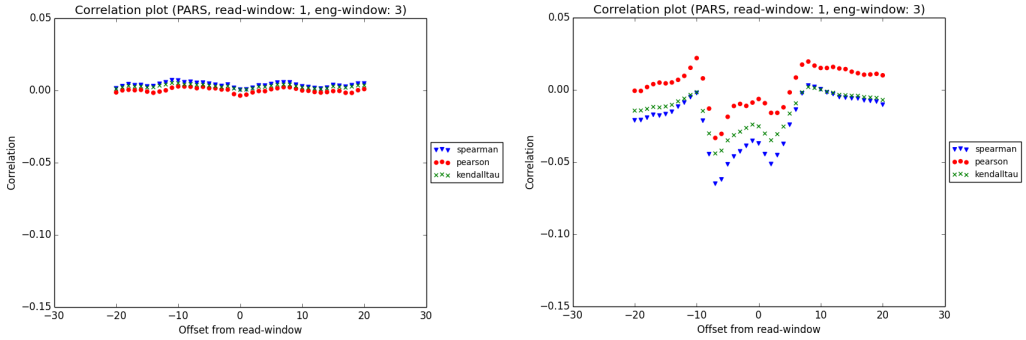

Figure B.4: Correlation plot for inverted PARS scores of the *Saccharomyces cerevisiae* mRNA sequences vs reads per codon at control data-set using *Pearson*, *Spearman* and *Kendall Tau* correlation measures. Left: randomly generated read data-set, no significant pattern in the correlation was observed. Right: wildtype mRNA-seq data-set, the data shows significant two dip pattern similar to the real ribosome profile data-set, but to a lesser extent.

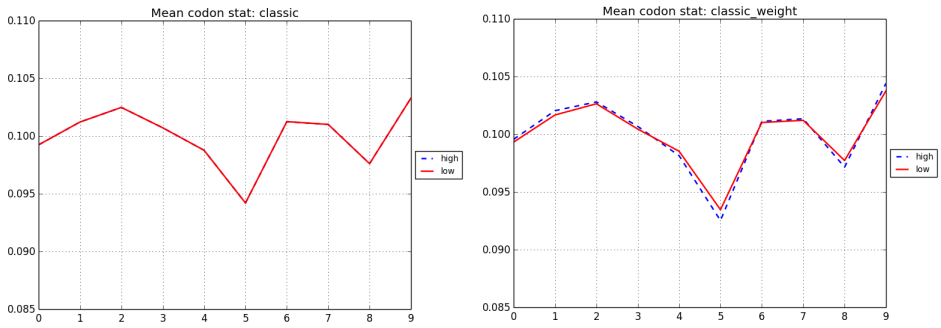

Figure B.5: RRT statistics for codons in the *high* and *low* group separated based on *classic* scoring method.

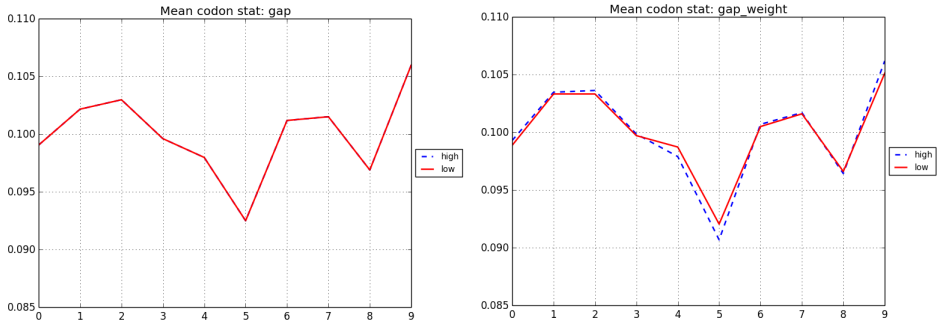

Figure B.6: RRT statistics for codons in the *high* and *low* group separated based on *gap* scoring method.

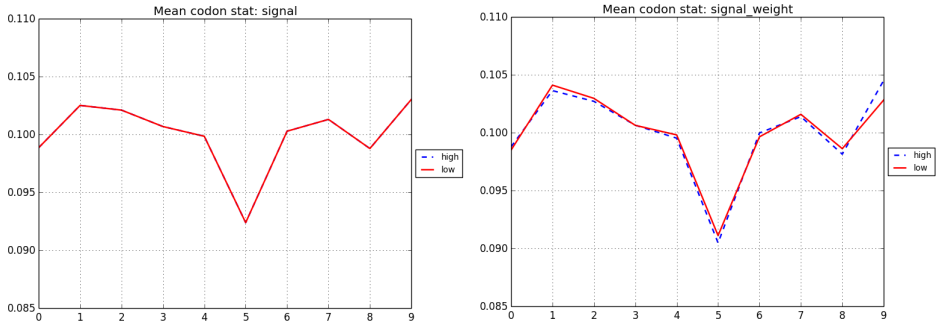

Figure B.7: RRT statistics for codons in the *high* and *low* group separated based on *signal* scoring method.

Bibliography

[1] Nih working definition of bioinformatics and computational biology. *Biomedical Information Science and Technology Initiative*, 2000.

[2] H. König, D. Frank, R. Heil, and C. Coenen. Synthetic genomics and synthetic biology applications between hopes and concerns. *Curr Genomics*, 2013.

[3] F. J. Isaacs, D. J. Dwyer, and J. J. Collins. Rna synthetic biology. *Nat Biotechnol.*, 24:545–54, 2006.

[4] Emma Abernathy, Karen Clyde, Rukhsana Yeasmin, Laurie T. Krug, Al Burlingame, Laurent Coscoy, and Britt Glaunsinger. Gammaherpesviral gene expression and virion composition are broadly controlled by accelerated mrna degradation. *PLoS Pathog*, 10, 2014.

[5] R. B. Weiss and J. F. Atkins. Molecular biology. translation goes global. *Science*, 334, 2011.

[6] T. Ikemura. Correlation between the abundance of escherichia coli transfer rnas and the occurrence of the respective codons in its protein genes: A proposal for a synonymous codon choice that is optimal for the e. coli translational system. *Journal of Molecular Biology*, 151(3):389 – 409, 1981.

[7] P. M. Sharp and W. H. Li. The codon adaptation index–a measure of directional synonymous codon usage bias, and its potential applications. *Nucleic Acids Res*, 15:1281–95, 1987.

[8] J. Plotkin and G. Kudla. Synonymous but not the same: the causes and consequences of codon bias. *Nature Reviews Genetics*, 12:32–42, 2011.

[9] J.R. Coleman, D. Papamichial, S. Skiena, B. Futcher, E. Wimmer, and S. Mueller. Virus attenuation by genome-scale changes in codon-pair bias. *Science*, 320:1784–1787, 2008.

[10] Gina Cannarozzi, Nicol N. Schraudolph, Mahamadou Faty, Peter von Rohr, Markus T. Friberg, Alexander C. Roth, Pedro Gonnet, Gaston Gonnet, and Yves Barral. A role for codon order in translation dynamics. *Cell*, 141:355–367, 2010.

[11] Rukhsana Yeasmin, Jesmin Jahan Tithi, Jeffrey Chen, and Steven Skiena. Designing autocorrelated genes. *ACM-BCB 2013*, 2013.

[12] Marilyn Kozak. Influences of mrna secondary structure on initiation by eukaryotic ribosomes. *Proc. Nati. Acad. Sci*, 83, 1986.

[13] G. Kudla, A.W. Murray, D. Tollervey, and J.B. Plotkin. Coding-sequence determinants of gene expression in escherichia coli. *Science*, 324:255–258, 2009.

[14] R. Yeasmin and S. Skiena. Designing rna secondary structures in coding regions. *International Symposium on Bioinformatics Research and Applications*, 7292, 2012.

[15] M. Zuker, D. H. Mathews, and D. H. Turner. Algorithms and thermodynamics for rna secondary structure prediction. 1999.

[16] J. Jaeger, D.H. Turner, and M. Zuker. Improved predictions of secondary structures for rna. *Proc. Natl. Acad. Sci. USA*, 86:7706–7710, 1989.

[17] I. Jr. Tinoco, P.N. Borer, B. Dengler, M.D. Levin, O.C. Uhlenbeck, D.M. Crothers, and J. Bralla. Improved estimation of secondary structure in ribonucleic acids. *Nat. New Biol.*, 246:40–41, 1973.

[18] I.L. Hofacker, W. Fontana, P.F. Stadler, and L.S. Bonhoeffer. Fast folding and comparison of rna secondary structures. *Monatshefte fur Chemie*, 125:167–188, 1994.

[Mfo] http://mfold.rna.albany.edu.

[Vie] http://rna.tbi.univie.ac.at.

[19] B. Cohen and S. Skiena. Natural selection and algorithmic design of mRNA. *J. Computational Biology*, 10:419–432, 2003.

[20] Barry Cohen and Steven Skiena. Optimizing rna secondary structure over all possible encodings of a given protein. *RECOMB*, 2000.

[21] Tamir Tuller, Yedael Y. Waldman, Martin Kupiec, and Eytan Ruppin. Translation efficiency is determined by both codon bias and folding energy. *Natl Acad Sci USA*, 2010.

[22] V. Sitaraman, P. Hearing, C. Ward, D. Gnatenko, Eckard Wimmer, S. Mueller, S. Skiena, and W. Bahou. Computationally designed adeno-associated virus (aav) rep 78 is efficiently maintained within an adenovirus vector. *Proc. National Academy of Sciences*, 108:14294–14299, 2011.

[23] S. Mueller, R. Coleman, D. Papamichail, C. Ward, A. Nimnual, B. Futcher, S. Skiena, and E. Wimmer. Live attenuated influenza vaccines by computer-aided rational design. *Nature Biotechnology*, 28, 2010.

[24] M. Zuker. Mfold web server for nucleic acid folding and hybridization prediction. *Nucleic Acids Res.*, 31:3406–3415, 2003.

[25] D.H. Mathews, J. Sabina, M. Zuker, and D.H. Turner. Expanded sequence dependence of thermodynamic parameters improves prediction of rna secondary structure. *J Mol Biol*, 288:911–940, 1999.

[26] M. Zuker and P. Stiegler. Optimal computer folding of large rna sequences using thermodynamics and auxiliary information. *Nucleic Acids Res.*, pages 133–48, 1981.

[27] JS. McCaskill. The equilibrium partition function and base pair binding probabilities for rna secondary structure. *Biopolymers*, 29:1105–19, 1990.

[28] S. Wuchty, Walter Fontana, I. L.H., and P. Schuster. Complete suboptimal folding of rna and the stability of secondary structures. *Biopolymers*, 49:145–165, 1999.

[29] B.A. Shapiro and K. Zhang. Comparing multiple rna secondary structures using tree comparisons. *Comput. Appl. Biosci.*, 6:309–318, 1990.

[30] Ivo Ludwig Hofacker. The rules of the evolutionary game for rna: A statistical characterization of the sequence to structure mapping in rna. *PhD thesis*, 1994.

[31] N. Dromi, A. Avihoo, and D. Barash. Reconstruction of natural rna sequences from rna shape, thermodynamic stability, mutational robustness, and linguistic complexity by evolutionary computation. *Biomol Struct Dyn*, 26:147–162, 2008.

[32] B.I. Dehiyat and S.L. Mayo. De novo protein design: Fully automated sequence selection. *Science*, 278:82–87, 1997.

[33] A. Busch and R. Backofen. Info-rna: a server for fast inverse rna folding satisfying sequence constraints. *Nucleic Acids Res.*, 2007.

[34] Assaf Avihoo, Alexander Churkin, and Danny Barash. Rnaexinv: An extended inverse rna folding from shape and physical attributes to sequences. *BMC Bioinformatics*, 2011.

[35] R. Durbin, S.R. Eddy, A. Krogh, and G. Mitchison. Biological sequence analysis: Probabilistic models of proteins and nucleic acids. *Cambridge University Press*, 1998.

[36] B. Knudsen and J. Hein. Rna secondary structure prediction using stochastic context-free grammars and evolutionary history. *Bioinformatics*, 15:446–454, 1999.

[37] R.D. Dowell and S.R. Eddy. Evaluation of several lightweight stochastic context-free grammars for rna secondary structure prediction. *BMC Bioinformatics*, 5, 2004.

[38] Chuong B. Do, Daniel A. Woods, and Serafim Batzoglou. Contrafold: Rna secondary structure prediction without physics-based models. *Bioinformatics*, 22:90–98, 2006.

[39] Mirela Andronescu, Anne Condon, Holger H. Hoos, David H. Mathews, and Kevin P. Murphy. Efficient parameter estimation for rna secondary structure prediction. *Bioinformatics*, 23, 2007.

[40] M. Andronescu. Computational approaches for rna energy parameter estimation. *PhD thesis, University of British Columbia, Vancouver, Canada*, 2008.

[41] Shay Zakov, Yoav Goldberg, Michael Elhadad, and Michal Ziv-Ukelson. Rich parameterization improves rna structure prediction. *RECOMB'11 Proceedings of the 15th Annual international conference on Research in computational molecular biology*, pages 546–562, 2011.

[42] Kishore J Doshi, Jamie J Cannone, Christian W Cobaugh, and Robin R Gutell. Evaluation of the suitability of free-energy minimization using nearest-neighbor energy parameters for rna secondary structure prediction. *BMC Bioinformatics*, 5:105, 2004.

[43] Nicholas Metropolis, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller, and Edward Teller. Equation of state by fast computing machines. *The Journal of Chemical Physics*, 21:1087–1092, 1953.

[44] Ruth Hershberg and Dmitri A. Petrov. Selection on codon bias. *Annu. Rev. Genet.*, 42, 2008.

[45] M. Welch, A. Villalobos, C. Gustafsson, and J. Minshull. Designing genes for successful protein expression. *Methods Enzymol.*, 498:43–66, 2011.

[46] C. Gustafsson, S. Govindarajan, and J. Minshull. Codon bias and heterologous protein expression. *Trends in biotechnology*, 22:346–53, 2004.

[47] T. F. Markus, G. Pedro, B. Yves, N. S. Nicol, and H. G. Gaston. Measures of codon bias in yeast, the trna pairing index and possible dna repair mechanisms. *WABI*, 2006.

[48] Philipp Bucher and Bernard M. E. Moret. Algorithms in bioinformatics. *6th international workshop, WABI 2006, Zurich, Switzerland*, 2006.

[49] G. Kudla, A. W. Murray, D. Tollervey, and J. B. Plotkin. Coding-sequence determinants of gene expression in escherichia coli. *Science*, 324, 2009.

[50] T. Tuller, A. Carmi, K. Vestsigian, S. Navon, Y. Dorfan, J. Zaborske, T. Pan, O. Dahan, I. Furman, and Y. Pilpel. An evolutionarily conserved mechanism for controlling the efficiency of protein translation. *Cell*, 141: 344–354, 2010.

[51] D. A. Drummond and C. O. Wilke. Mistranslation-induced protein misfolding as a dominant constraint on coding-sequence evolution. *Cell*, 134:341–352, 2008.

[52] T. Ikemura. Codon usage and trna content in unicellular and multicellular organisms. *Mol. Biol. Evol.*, 2:13–34, 1985.

[53] P.M. Sharp, M. Stencio, J.F. Peden, and A.T. Lloyd. Codon usage: mutational bias, translational selection, or both? *Biochem. Soc. Trans.*, 21:835841, 1993.

[54] H. Dong, L. Nilsson, and C. G. Kurland. Co-variation of trna abundance and codon usage in escherichia coli at different growth rates. *J Mol Biol*, 260:649–63, 1996.

[55] A. Eyre-Walker and M. Bulmer. Synonymous substitution rates in enterobacteria. *Genetics*, 140:1407–1412, 1995.

[56] Chen Xu, Jing Dong, Chunfa Tong, Xindong Gong, Qiang Wen, and Qiang Zhuge. Analysis of synonymous codon usage patterns in seven different citrus species. *Evolutionary bioinformatics online*, 9:215, 2013.

[57] Wenfeng Qian, Jian-Rong Yang, Nathaniel M Pearson, Calum Maclean, and Jianzhi Zhang. Balanced codon usage optimizes eukaryotic translational efficiency. *PLoS genetics*, 8(3), 2012.

[cel] http://genome-www.stanford.edu/cellcycle/data/rawdata/combined.txt.

[58] R. J. Cho, M. J. Campbell, E. A. Winzeler, L. Steinmetz, A. Conway, L. Wodicka, T. G. Wolfsberg, A. E. Gabrielian, D. Landsman, D. J. Lockhart, and R. W. Davis. A genome-wide transcriptional analysis of the mitotic cell cycle. *Cell*, 2:65–67, 1998.

[59] P. Spellman, G. Sherlock, M. Zhang, V. Iyer, K. Anders, M. Eisen, P. Brown, D. Botstein, and B. Futcher. Comprehensive identification of cell cycle-regulated genes of the yeast saccharomyces cerevisiae by microarray hybridization. *Cell*, 9, 1998.

[60] D. Lipson, T. Raz, A. Kieu1, D. R Jones, E. Giladi, E. Thayer, J. F Thompson, S. Letovsky, P. Milos, and M. Causey. Quantification of the yeast transcriptome by single-molecule sequencing. *Nature Biotechnology*, 27:652–658, 2009.

[61] Q. Tian, S.B. Stepaniants, M. Mao, L. Weng, M.C. Feetham, and M.J. Doyle. Integrated genomic and proteomic analyses of gene expression in mammalian cells. *Molecular & Cellular Proteomics*, 3(10), 2004.

[62] M. Siwiak and P Zielenkiewicz. Integrated genomic and proteomic analyses of gene expression in mammalian cells. *A Comprehensive, Quantitative, and Genome-Wide Model of Translation*, 6(7), 2010.

[63] Donna K. Slonim and Itai Yanai. Getting started in gene expression microarray analysis. *PLoS Computational Biology*, 2009.

[64] S. Covarrubias, M. M. Gaglia, G. R. Kumar, W. Wong, A. O. Jackson, and B. A. Glaunsinger. Coordinated destruction of cellular messages in translation complexes by the gammaherpesvirus host shutoff factor and the mammalian exonuclease xrn1. *PLoS Pathog*, 2011.

[65] Benson Yee Hin Cheng, Jizu Zhi, Alexis Santana, Sohail Khan, Eduardo Salinas, J. Craig Forrest, Yueting Zheng, Shirin Jaggi, Janet Leatherwood, and Laurie T. Krug. Tiled microarray identification of novel viral transcript structures and distinct transcriptional profile during two modes of productive murine gammaherpesvirus 68 infection. *J. Virol.*, 86, 2012.

[66] H. W. Virgin, P. Latreille, P. Wamsley, K. Hallsworth, K. E. Weck, A. J. Dal Canto, and S. H. Speck. Complete sequence and genomic analysis of murine gammaherpesvirus 68. *J. Virol.*, 71, 1997.

[67] Hendrik J. M. de Jonge, Rudolf S. N. Fehrmann, Eveline S. J. M. de Bont, Robert M. W. Hofstra, Frans Gerbens, Willem A. Kamps, Elisabeth G. E. de Vries, Ate G. J. van der Zee, Gerard J. te Meerman, and Arja ter Elst. Evidence based selection of housekeeping genes. *PLoS ONE*, 2007.

[68] G. K. Smyth. Linear models and empirical bayes methods for assessing differential expression in microarray experiments. *Statistical Applications in Genetics and Molecular Biology*, 3, 2004.

[69] Justin M. Richner, Karen Clyde, Andrea C. Pezda, Benson Yee Hin Cheng, Tina Wang, G. Renuka Kumar, Sergio Covarrubias, Laurent Coscoy, and Britt Glaunsinger. Global mrna degradation during lytic gammaherpesvirus infection contributes to establishment of viral latency. *PLoS Pathog*, 7, 2011.

[70] A. D. Kwong and N. Frenkel. The herpes simplex virus virion host shutoff function. *J Virol*, 63, 1989.

[71] L. I. Strelow and D. A. Leib. Role of the virion host shutoff (vhs) of herpes simplex virus type 1 in latency and pathogenesis. *J Virol*, 69, 1995.

[72] B. Glaunsinger and D. Ganem. Lytic kshv infection inhibits host gene expression by accelerating global mrna turnover. *Mol Cell*, 13, 2004.

[73] S. Covarrubias, J. M. Richner, K. Clyde, Y. J. Lee, and B. A. Glaunsinger. Host shutoff is a conserved phenotype of gammaherpesvirus infection and is orchestrated exclusively from the cytoplasm. *J Virol*, 83, 2009.

[74] S. Covarrubias, M. M. Gaglia, G. R. Kumar, W. Wong, A. O. Jackson, and B. Glaunsinger. Coordinated destruction of cellular messages in translation complexes by the gammaherpesvirus host shutoff factor and the mammalian exonuclease xrn1. *PLoS Pathog*, 7, 2011.

[75] M. M. Gaglia, S. Covarrubias, W. Wong, and B. A. Glaunsinger. A common strategy for host rna degradation by divergent viruses. *J Virol*, 86, 2012.

[76] H. G. Krausslich, M. J. Nicklin, H. Toyoda, D. Etchison, and E. Wimmer. Poliovirus proteinase 2a induces cleavage of eucaryotic initiation factor 4f polypeptide p220. *J Virol*, 61, 1987.

[77] M. Zamora, W. E. Marissen, and R. E. Lloyd. Multiple eif4gi-specific protease activities present in uninfected and poliovirus-infected cells. *J Virol*, 76, 2002.

[78] K. S. Sciabica, Q. J. Dai, and R. M. Sandri-Goldin. Icp27 interacts with srpk1 to mediate hsv splicing inhibition by altering sr protein phosphorylation. *EMBO J*, 22, 2003.

[79] M. D. Koffa, J. B. Clements, E. Izaurralde, S. Wadd, S. A. Wilson, I. W. Mattaj, and S. Kuersten. Herpes simplex virus icp27 protein provides viral mrnas with access to the cellular mrna export pathway. *EMBO J*, 20, 2001.

[80] I. H. Chen, K. S. Sciabica, and R. M. Sandri-Goldin. Icp27 interacts with the rna export factor aly/ref to direct herpes simplex virus type 1 intronless mrnas to the tap export pathway. *J Virol*, 76, 2002.

[81] C. Huang, K. G. Lokugamage, J. M. Rozovics, K. Narayanan, B. L. Semler, and S. Makino. Sars coronavirus nsp1 protein induces template-dependent endonucleolytic cleavage of mrnas: viral mrnas are resistant to nsp1-induced rna cleavage. *PLoS Pathog*, 7, 2011.

[82] K. Clyde and B. A. Glaunsinger. Deep sequencing reveals direct targets of gammaherpesvirus-induced mrna decay and suggests that multiple mechanisms govern cellular transcript escape. *PLoS One*, 6, 2011.

[83] S. Hutin, Y. Lee, and B. A. Glaunsinger. An rna element in human interleukin 6 confers escape from degradation by the gammaherpesvirus sox protein. *J Virol*, 87, 2013.

[84] J. Gardin, R. Yeasmin, A. Yurovsky, Y. Cai, S. Skiena, and B. Futcher. Measurement of average decoding rates of the 61 sense codons in vivo. *eLife*, 2014.

[85] N. T. Ingolia, S. Ghaemmaghami, J. R. S. Newman, and J. S. Weissman. Genome-wide analysis in vivo of translation with nucleotide resolution using ribosome profiling. *Science*, 324:218–223, 2009.

[86] G. W. Li, E. Oh, and J. S. Weissman. The anti-shine-dalgarno sequence drives translational pausing and codon choice in bacteria. *Nature*, 484: 538–41, 2012.

[87] L. F. Lareau, D. H. Hite, G. J. Hogan, and P. O. Brown. Distinct stages of the translation elongation cycle revealed by sequencing ribosome-protected mrna fragments. *eLife*, 3, 2014.

[88] J. L. Bennetzen and B. D. Hall. Codon selection in yeast. *J Biol Chem*, 257:3026–31, 1982.

[89] M. Bulmer. Coevolution of codon usage and transfer rna abundance. *Nature*, 325:728–30, 1987.

[90] W. M. Fitch. Is there selection against wobble in codon-anticodon pairing? *Science*, 194:1173–4, 1976.

[91] M. Hasegawa, T. Yasunaga, and T. Miyata. Secondary structure of ms2 phage rna and bias in code word usage. *Nucleic Acids Res*, 7:2073–9, 1979.

[92] D. J. Lipman and W. J. Wilbur. Contextual constraints on synonymous codon choice. *J Mol Biol*, 163:363–76, 1983.

[93] T. Miyata, H. Hayashida, T. Yasunaga, and T. Yasunaga. The preferential codon usages in variable and constant regions of immunoglobulin genes are quite distinct from each other. *Nucleic Acids Res*, 7:2431–8, 1979.

[94] P. M. Sharp and W. H. Li. An evolutionary perspective on synonymous codon usage in unicellular organisms. *J Mol Evol*, 24:28–38, 1986.

[95] B. Maertens, A. Spriestersbach, U. Von Groll, U. Roth, J. Kubicek, M. Gerrits, M. Graf, M. Liss, D. Daubert, R. Wagner, and F. Schafer. Gene optimization mechanisms: a multi-gene study reveals a high success rate of full-length human proteins expressed in escherichia coli. *Protein Sci*, 19:1312–26, 2010.

[96] N. A. Burgess-Brown, S. Sharma, F. Sobott, C. Loenarz, U. Oppermann, and O. Gileadi. Codon optimization can improve expression of human genes in escherichia coli: A multi-gene study. *Protein Expr Purif*, 59: 94–102, 2008.

[97] N. T. Ingolia, L. F. Lareau, and J. S. Weissman. Ribosome profiling of mouse embryonic stem cells reveals the complexity and dynamics of mammalian proteomes. *Cell.*, 147:789–802, 2011.

[98] C. A. Charneski and L. D. Hurst. Positively charged residues are the major determinants of ribosomal velocity. *PLoS Biol*, 11, 2013.

[99] O. Brandman, J. Stewart-Ornstein, D. Wong, A. Larson, C. C. Williams, G. W. Li, S. Zhou, D. King, P. S. Shen, J. Weibezahn, J. G. Dunn, S. Rouskin, T. Inada, A. Frost, and J. S. Weissman. A ribosome-bound quality control complex triggers degradation of nascent peptides and signals translation stress. *Cell*, 151:1042–54, 2012.

[100] Nicholas T. Ingolia, Gloria A. Brar, Silvia Rouskin, Anna M. McGeachy, and Jonathan S. Weissman. The ribosome profiling strategy for monitoring translation in vivo by deep sequencing of ribosome-protected mrna fragments. *Nat Protoc.*, 7:15341550, 2012.

[101] Y. Arava, Y. Wang, J. D. Storey, C. L. Liu, P. O. Brown, and D. Herschlag. Genome-wide analysis of mrna translation profiles in saccharomyces cerevisiae. *Proc Natl Acad Sci*, 100:3889–94, 2003.

[102] P. Sampath, D. K. Pritchard, L. Pabon, H. Reinecke, S. M. Schwartz, D. R. Morris, and C. E. Murry. A hierarchical network controls protein translation during murine embryonic stem cell self-renewal and differentiation. *Cell Stem Cell.*, 8:448–60, 2008.

[103] Nicholas T. Ingolia. Ribosome profiling: new views of translation, from single codons to genome scale. *Nature Reviews Genetics*, 15:205–213, 2014.

[104] Alexandra Dana and Tamir Tuller. Determinants of translation elongation speed and ribosomal profiling biases in mouse embryonic stem cells. *PLoS Comput Biol*, 8, 2012.

[105] A. T. Lamm, M. R. Stadler, H. Zhang, J. I. Gent, and A. Z. Fire. Multimodal rna-seq using single-strand, double-strand, and circligase-based capture yields a refined and extended description of the c. elegans transcriptome. *Genome Res*, 21:265–75, 2011.

[106] C. A. Raabe, T. H. Tang, J. Brosius, and T. S. Rozhdestvensky. Biases in small rna deep sequencing data. *Nucleic Acids Res*, 42:1414–26, 2014.

[107] T. J. Jackson, R. V. Spriggs, N. J. Burgoyne, C. Jones, and A. E. Willis. Evaluating bias-reducing protocols for rna sequencing library preparation. *BMC Genomics*, 15, 2014.

[108] B. Futcher, G. I. Latter, P. Monardo, C. S. McLaughlin, and J. I. Garrels. A sampling of the yeast proteome. *Mol Cell Biol.*, 19, 1999.

[109] M. Johansson, K. W. Ieong, S. Trobro, P. Strazewski, J. Aqvist, M. Y. Pavlov, and M. Ehrenberg. ph-sensitivity of the ribosomal peptidyl transfer reaction dependent on the identity of the a-site aminoacyl-trna. *Proc Natl Acad Sci USA*, 108:79–84, 2011.

[110] H. Muto and K. Ito. Peptidyl-prolyl-trna at the ribosomal p-site reacts poorly with puromycin. *Biochem Biophys Res Commun.*, 366:1043–1047, 2008.

[111] Michael Y. Pavlov, Richard E. Watts, Zhongping Tan, Virginia W. Cornish, Mans Ehrenberg, and Anthony C. Forster. Slow peptide bond formation by proline and other n-alkylamino acids in translation. *Proc Natl Acad Sci*, 106, 2009.

[112] L. K. Doerfel, I. Wohlgemuth, C. Kothe, F. Peske, H. Urlaub, and M. V. Rodnina. Ef-p is essential for rapid synthesis of proteins containing consecutive proline residues. *Science*, 339:85–8, 2013.

[113] L. Peil, A. L. Starosta, J. Lassak, G. C. Atkinson, K. Virumae, M. Spitzer, T. Tenson, K. Jung, J. Remme, and D. N. Wilson. Distinct xppx sequence motifs induce ribosome stalling, which is rescued by the translation elongation factor ef-p. *Proc Natl Acad Sci USA*, 110: 15265–70, 2013.

[114] S. Ude, J. Lassak, A. L. Starosta, T. Kraxenberger, D. N. Wilson, and K. Jung. Translation elongation factor ef-p alleviates ribosome stalling at polyproline stretches. *Science*, 339:82–5, 2013.

[115] T. Ikemura. Correlation between the abundance of yeast transfer rnas and the occurrence of the respective codons in protein genes. differences in synonymous codon choice patterns of yeast and escherichia coli with reference to the abundance of isoaccepting transfer rnas. *J Mol Biol*, 158:573–97, 1982.

[116] A. C. Roth. Decoding properties of trna leave a detectable signal in codon usage bias. *Bioinformatics*, 28, 2012.

[117] T. Daviter, K. B. Gromadski, and M. V. Rodnina. The ribosome's response to codon-anticodon mismatches. *Biochimie*, 88, 2006.

[118] K. B. Gromadski, T. Daviter, and M. V. Rodnina. A uniform response to mismatches in codon-anticodon complexes ensures ribosomal fidelity. *Mol Cell*, 21, 2006.

[119] P. K. Khade and S. Joseph. Messenger rna interactions in the decoding center control the rate of translocation. *Nat Struct Mol Biol*, 18, 2011.

[120] Y. P. Semenkov, M. V. Rodnina, and W. Wintermeyer. Energetic contribution of trna hybrid state formation to translocation catalysis on the ribosome. *Nat Struct Biol*, 7, 2000.

[121] J. Lu and C. Deutsch. Electrostatics in the ribosomal tunnel modulate chain elongation rates. *J Mol Biol*, 384, 2008.

[122] G. Blaha, G. Gurel, S. J. Schroeder, P. B. Moore, and T. A. Steitz. Mutations outside the anisomycin-binding site can make ribosomes drug-resistant. *J Mol Biol*, 379, 2008.

[123] J. L. Hansen, P. B. Moore, and T. A. Steitz. Structures of five antibiotics bound at the peptidyl transferase center of the large ribosomal subunit. *J Mol Biol*, 330, 2003.

[124] P. Bieling, M. Beringer, S. Adio, and M. V. Rodnina. Peptide bond formation does not involve acid-base catalysis by ribosomal residues. *Nat Struct Mol Biol*, 13:423–8, 2006.

[125] N. Demeshkina, L. Jenner, E. Westhof, M. Yusupov, and G. Yusupova. A new understanding of the decoding principle on the ribosome. *Nature*, 484:256–9, 2012.

[126] X. Zeng, J. Chugh, A. Casiano-Negroni, H. M. Al-Hashimi, and 3RD Brooks, C. L. Flipping of the ribosomal a-site adenines provides a basis for trna selection. *J Mol Biol*, 426, 2014.

[127] M. V. Rodnina. The ribosome as a versatile catalyst: reactions at the peptidyl transferase center. *Curr Opin Struct Biol*, 23, 2013.

[128] Y. S. Polikanov, T. A. Steitz, and C. A. Innis. A proton wire to couple aminoacyl-trna accommodation and peptide-bond formation on the ribosome. *Nat Struct Mol Biol*, 21, 2014.

[129] J. Zhou, L. Lancaster, J. P. Donohue, and H. F. Noller. How the ribosome hands the a-site trna to the p site during ef-g-catalyzed translocation. *Science*, 345, 2014.

[130] P. M. Petrone, C. D. Snow, D. Lucent, and V. S. Pande. Side-chain recognition and gating in the ribosome exit tunnel. *Proc Natl Acad Sci USA*, 105, 2008.

[131] J. Lu, Z. Hua, W. R. Kobertz, and C. Deutsch. Nascent peptide side chains induce rearrangements in distinct locations of the ribosomal tunnel. *J Mol Biol*, 411, 2011.

[132] S. G. Andersson and C. G. Kurland. Codon preferences in free-living microorganisms. *Microbiol Rev*, 54, 1990.

[133] D. Chu, E. Kazana, N. Bellanger, T. Singh, M. F. Tuite, and T. Von Der Haar. Translation elongation can control translation initiation on eukaryotic mrnas. *EMBO J*, 33, 2014.

[134] U. Berndt, S. Oellerer, Y. Zhang, A. E. Johnson, and S. Rospert. A signal-anchor sequence stimulates signal recognition particle binding to ribosomes from inside the exit tunnel. *Proc Natl Acad Sci USA*, 106, 2009.

[135] U. Raue, S. Oellerer, and S. Rospert. Association of protein biogenesis factors at the yeast ribosomal tunnel exit is affected by the translational status and nascent polypeptide sequence. *J Biol Chem*, 282, 2007.

[136] J. C. Darnell, S. J. Van Driesche, C. Zhang, K. Y. Hung, A. Meleand, C. E. Fraser, E. F. Stone, C. Chen, J. J. Fak, S. W. Chi, D. D. Licatalosi, J. D. Richter, and R. B. Darnell. Fmrp stalls ribosomal translocation on mrnas linked to synaptic function and autism. *Cell.*, 146:247–61, 2011.

[137] D. R. Morris and A. P. Geballe. Review upstream open reading frames as regulators of mrna translation. *Mol Cell Biol.*, 20:8635–42, 2000.

[138] C. Kimchi-Sarfaty, J. M. Oh, I. W. Kim, Z. E. Sauna, A. M. Calcagno, S. V. Ambudkar, and M. M. Gottesman. A silent polymorphism in the mdr1 gene changes substrate specificity. *Science.*, 315:525–8, 2007.

[139] G. Zhang, M. Hubalewska, and Z. Ignatova. Transient ribosomal attenuation coordinates protein synthesis and co-translational folding. *Nat Struct Mol Biol.*, 16:274–80, 2009.

[140] M. Mariappan, X. Li, S. Stefanovic, A. Sharma, A. Mateja, R. J. Keenan, and R. S. Hegde. A ribosome-associating factor chaperones tail-anchored membrane proteins. *Nature.*, 466:1120–4, 2010.

[141] K. Yanagitani, Y. Kimata, H. Kadokura, and K. Kohno. Translational pausing ensures membrane targeting and cytoplasmic splicing of xbp1u mrna. *Science.*, 331:586–9, 2011.

[142] B. Irwin, J. D. Heck, and G. W. Hatfield. Codon pair utilization biases influence translational elongation step times. *J Biol Chem.*, 270:22801–6, 1995.

[143] O. Namy, S. J. Moran, D. I. Stuart, R. J. Gilbert, and I. Brierley. A mechanical explanation of rna pseudoknot function in programmed ribosomal frameshifting. *Nature.*, 441:244–7, 2006.

[144] H. Nakatogawa and K. Ito. The ribosomal exit tunnel functions as a discriminating gate. *Cell.*, 108:629–36, 2002.

[145] Simon Anders, Paul Theodor Pyl, and Wolfgang Huber. Htseq - a python framework to work with high-throughput sequencing data. *bioRxiv preprint*, 2014.

[146] M. Yarus and L. S. Folley. Sense codons are found in specific contexts. *J. Mol. Biol.*, 182, 1985.

[147] T. Taniguchi and C. Weissmann. Inhibition of qbeta rna 70s ribosome initiation complex formation by an oligonucleotide complementary to the 3' terminal region of e. coli 16s ribosomal rna. *Nature*, 275:770772, 1978.

[148] L. Bossi and J. R. Ruth. The influence of codon context on genetic code translation. *Nature*, 286, 1980.

[149] E. J. Murgola, F. T. Pagel, and K. A. Hijazi. Codon context effects in missense suppression. *J. Mol. Biol.*, 175, 1984.

[150] G. A. Gutman and G. W. Hatfield. Non-random utilization of codon pairs in escherichia coli. *Proc. Natl Acad. Sci.*, 86, 1989.

[151] B. Irwin, J. D. Heck, and G. W. Hatfield. Codon pair utilization biases influence translational elongation step times. *J Biol Chem.*, 1995.

[152] D. Smith and M Yarus. trna-trna interactions within cellular ribosomes. *Proc. Natl Acad. Sci.*, 86:43974401, 1989.

[153] M. Yarus and J. Curran. Transfer rna in protein synthesis. *CRC Press*, page 319365, 1992.

[154] K. H. Nierhaus, J. Wadzack, N. Burkhardt, R. Junemann, W. Meerwinck, R. Willumeit, and H. B. Stuhrmann. Structure of the elongating ribosome: arrangement of the two trnas before and after translocation. *Proc. Natl Acad. Sci.*, 1998.

[155] J. R. Buchan, L. S. Aucott, and I. Stansfield. trna properties help shape codon pair preferences in open reading frames. *Nucleic Acids Research*, 2006.

[156] Svetlana Boycheva, Georgi Chkodrov, and Ivan Ivanov. Codon pairs in the genome of escherichia coli. *Bioinformatics*, 19:987–998, 2003.

[157] G. W. Hatfield and G. A. Gutman. Transfer rna in protein synthesis. *CRC Press*, 86:157190, 1989.

[158] P. M. Sharp and W. H. Li. Codon usage in regulatory genes in escherichia coli does not reflect selection for 'rare codons. *Nucleic Acids Research*, 1986.

[159] T. Ikemura. Transfer rna in protein synthesis. *CRC Press*, page 113124, 1992.

[160] R. B. Loftfield and D. Vanderjagt. The frequency of errors in protein biosynthesis. *Biochem. J.*, 1972.

[161] I. Stansfield, K. M. Jones, P. Herbert, A. Lewendon, W. V. Shaw, and M. F. Tuite. Missense translation errors in saccharomyces cerevisiae. *J. Mol. Biol.*, 1998.

[162] L. Boe. Translational errors as the cause of mutations in escherichia coli. *Mol. Gen. Genet.*, 1992.

[163] P. J. Farabaugh. Programmed translational frameshifting. *Annu Rev Genet.*, 1996.

[164] C. G. Kurland. Translational accuracy and the fitness of bacteria. *Annu Rev Genet*, 1992.

[165] A. A. Shah, M. C. Giddings, J. B. Parvaz, R. F. Gesteland, and Ivanov I. P. Atkins, J. F. Computational identification of putative programmed translational frameshift sites. *Bioinformatics*, 2002.

[166] L. Green, C. H. Kim, C. Bustamante, and I. Tinoco. Characterization of the mechanical unfolding of rna pseudoknots. *J. Mol. Biol.*, 375, 2008.

[Wik] http://en.wikipedia.org/wiki/Translational_frameshift.

[167] M. Kozak. Regulation of translation via mrna structure in prokaryotes and eukaryotes. *Gene*, 361, 2005.

[168] L. Katz and C. B. Burge. Widespread selection for local rna secondary structure in coding regions of bacterial genes. *Genome Res.*, 2003.

[169] Michael Kertesz, Yue Wan, Elad Mazor, John L. Rinn, Robert C. Nutter, Howard Y. Chang, and Eran Segal. Genome-wide measurement of rna secondary structure in yeast. *Nature*, 467:103107, 2010.

[170] D. Chu, D. J. Barnes, and T. von der Haar. The role of trna and ribosome competition in coupling the expression of different mrnas in saccharomyces cerevisiae. *Nucleic Acids Research*, 2011.

[171] E. P. Rocha. Codon usage bias from trna's point of view: redundancy, specialization, and efficient decoding for translation optimization. *Genome Res.*, 2004.

[172] M. dos Reis, L. Wernisch, and R. Savva. Unexpected correlations between gene expression and codon usage bias from microarray data for the whole escherichia coli k-12 genome. *Nucleic Acids Res.*, 2003.

[173] T. E. Gorochowski, Z. Ignatova, R. A. L. Bovenberg, and J. A. Roubos. Trade-offs between trna abundance and mrna secondary structure support smoothing of translation elongation rate. *Nucl. Acids Res.*, 2015.

[174] B. S. Negrutskii and M. P. Deutscher. Channeling of aminoacyl-trna for protein synthesis in vivo. *Proc Natl Acad Sci USA*, 1991.

[175] R. Stapulionis and M. P. Deutscher. A channeled trna cycle during mammalian protein synthesis. *Proc Natl Acad Sci USA*, 1995.

[176] M. Kaminska, S. Havrylenko, P. Decottignies, P. Le Maréchal, B. Negrutskii, and M. Mirande. Dynamic organization of aminoacyl-trna synthetase complexes in the cytoplasm of human cells. *J. Biol. Chem.*, 2009.

[177] Z. Q. Shao, Y. M. Zhang, X. Y. Feng, B. Wang, and J. Q. Chen. Synonymous codon ordering: A subtle but prevalent strategy of bacteria to improve translational efficiency. *PLoS One*, 2012.

[178] V. Godinic-Mikulcic, J. Jaric, B. J. Greber, V. Franke, V. Hodnik, G. Anderluh, N. Ban, and I. Weygand-Durasevic. Archaeal aminoacyl-trna synthetases interact with the ribosome to recycle trnas. *Nucleic Acids Res.*, 2014.

[179] S. R. McGuffee and A. H. Elcock. Diffusion, crowding & protein stability in a dynamic molecular model of the bacterial cytoplasm. *PLoS Comput. Biol.*, 2010.

[180] T. F. Knight. Engineering novel life. *Mol. Syst. Biol.*, 13, 2005.

[181] W. Szybalski and A. Skalka. Nobel prizes and restriction enzymes. *Gene*, 4, 1978.

[182] D. Endy. Foundations for engineering biology. *Nature*, 438, 2005.

[183] Kosuri S. Chan, L. Y. and D. Endy. Refactoring bacteriophage. *Mol. Syst. Biol.*, 2005.

[184] E. Wimmer, S. Mueller, T. M. Tumpey, and J. K. Taubenberger. Synthetic viruses: a new opportunity to understand and prevent viral disease. *Nat. Biotechnol.*, 2009.

[185] Z. Xie, L. Wroblewska, L. Prochazka, R. Weiss, and Y. Benenson. Multi-input rnai-based logic circuit for identification of specific cancer cells. *Science*, 2011.

[186] W. Weber, R. Schoenmakers, B. Keller, M. Gitzinger, T. Grau, M. D. Baba, P. Sander, and M. Fussenegger. A synthetic mammalian gene circuit reveals antituberculosis compounds. *Proc Natl Acad Sci USA*, 2008.

[187] A. S. Khalil and J. J. Collins. Synthetic biology: applications come of age. *Nat Rev Genet*, 2010.