Minimizing Interference due to Genetic Manipulation

By

Jerry S. Wang

B.S. Electrical Engineering and Computer Science, M.I.T. 2012

Submitted to the Department of Electrical Engineering and Computer Science
In Partial Fulfillment of the Requirements for the Degree of
Master of Engineering and Computer Science
At the Massachusetts Institute of Technology
May 2013
(June 2013)

Author: _____
Department of Electrical Engineering and Computer Science
May 24, 2013

Certified by: _____
Assistant Prof. Timothy K. Lu, Thesis Supervisor
May 24, 2013

Accepted by: _____
Prof. Dennis M. Freeman, Chairman, Masters of Engineering Thesis Committee

Abstract

Synthetic gene networks are the functional genetic 'programs' that will form the basis of increasingly sophisticated engineered organisms, cells and tissues. Important factors to consider in biologically engineering circuits are modularity and orthogonality of the components. We explore these factors experimentally by using TALEs as a synthetic transcription factor. We also designed an algorithm to find optimal locations to insert synthetic gene networks into the cell so that interference is minimized and orthogonality is maximized. Finally, we developed a method for encrypting and decrypting these genetic 'programs' for the protection of intellectual property. Together these projects explore ways of minimizing unwanted effects due to genetic manipulation.

Preface and Acknowledgements

This document gives a summary of my work while completing the Masters of Engineering degree during the 2012-2013 academic year at the Massachusetts Institute of Technology. All work was conducted in the Lu Lab in building NE47 on MIT campus under the supervision of Professor Timothy Lu.

My background is a B.S in Electrical Engineering and Computer Science with a Minor in Biology. Working in the Lu Lab was my first experience working in a wet laboratory environment. The initial transition was tough but thanks to the help of my lab mates, I was able to get up to speed and proficient in lab techniques. The projects in this thesis all have future components that need to be completed; my current progress has opened new, more interesting questions and directions and it was not possible to answer and explore them all given the timeline of the M.Eng degree.

I would like to thank everyone in the Lu Lab for providing me with help throughout the year. In particular, I would like to thank Allen Cheng, who patiently taught me lab techniques and acted as a mentor, Sam Perli and Fahim Farzadfrrd, with whom I consulted incessantly while working on my experimental project, and Piro Siuti and Oliver Purcell, with whom I collaborated with on the computational projects. The expertise and helpful nature of those people made my M.Eng an enjoyable and rewarding experience. Finally I would like to thank Professor Tim Lu, without whom, none of this would have been possible. Professor Lu provided great guidance and was an invaluable source of inspiration.

Table of Contents

Contents Page

Overview

The overarching theme of my M.Eng work was answering the question: How can we manipulate the genome to do what we want to do without unintended affects? I undertook three different projects. One was mostly experimental and the other two were mostly computational with experimental validation as future components.

The first project explores the TALE-DNA interactions. It uses a robust TALE vector we built to bind to targeted as well as random sequences in the genome and reporter constructs. Our goal was to use the data generated to create an orthogonality matrix and quantify how specific TALEs were and how we can design TALEs to be as orthogonal as possible.

The second project was developing a systematic way to find the best insertion sites for synthetic constructs into the E. coli genome. We wanted to find integration sites such that the integrated component did not affect surrounding genes and the genome as a whole as well as minimizing the effects that the genes had on the integrated component. The algorithm to find these sites can be easily applied to other organisms. The resulting sites we found will need to be experimentally validated and statistically verified to be better than previous known integration sites.

The final project was a new idea into how to protect genetic material that we called Genetic Encryption. It is a way of scrambling the sequence or network topology of a construct such that it would be impossible to know what the original sequence was without unscrambling or decrypting with a "key". The method will have to be experimentally verified to ensure that it works in practice and not just in theory.

Modeling Specificity of TALE-DNA interactions

Introduction

The cell can be seen as an integrated device made of several thousand types of interacting proteins. Cells live in complex environments and need to sense different signals such as temperature, pressure, and signals from other molecules [10] and produce different amounts of certain proteins as a response. Proteins can interact with DNA to accomplish a wide variety of these cellular processes, including DNA replication and repair. The key component to mediate all the interactions and responses are transcription factors. Transcription factors are usually designed to rapidly transition between active and inactive states based on the environmental signals. Active transcriptions factors can then bind to the DNA to regulate the rate at which specific targeted genes are produced (figure 1).

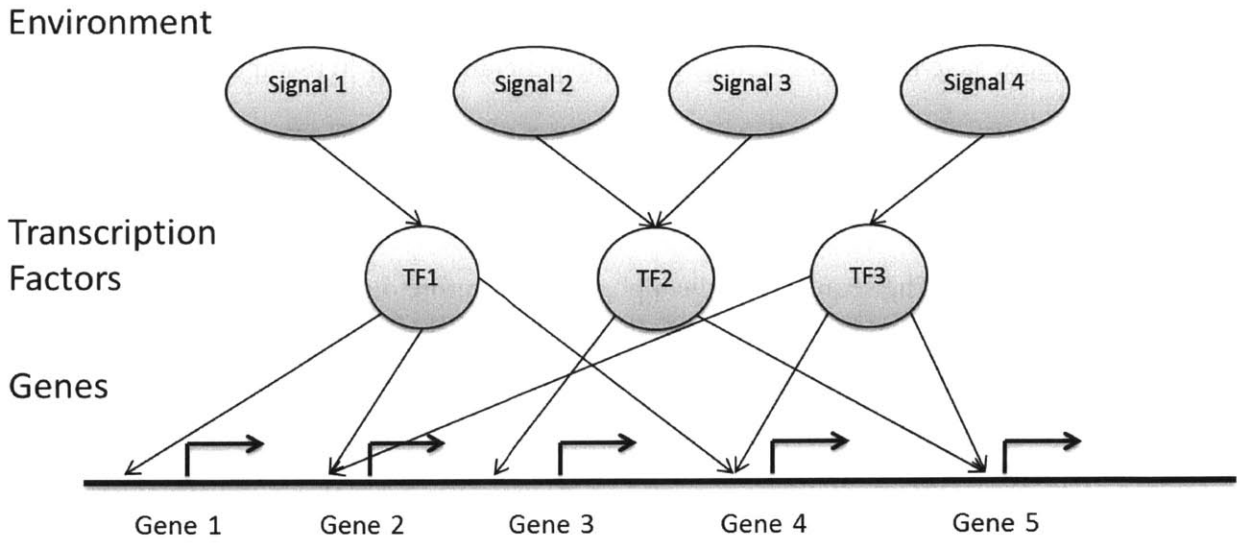


Figure 1: The mapping between environmental signals, transcription factors inside the cells, and the genes that they regulate. Transcription factors are the layers separating the signals from the genetic response. It is important to note that each factor can be activated or repressed by many or no signals and can activate or repress many or no genetic responses.

Eukaryotic transcription factors perform more complex and combinatorial functions within transcriptional networks than their prokaryotic counterparts, which is my most previous studies

have been performed in yeast. Several types of programmable DNA binding proteins have recently been devised among which are the zinc finger proteins (ZFs) and the Transcription-Activator like Effectors (TALEs) [1-5]. The DNA binding proteins can be fused to other kinds of proteins, such as activation domains, repression domains, and nucleases to specifically target a certain gene for a certain type of change in activity (figure 2).

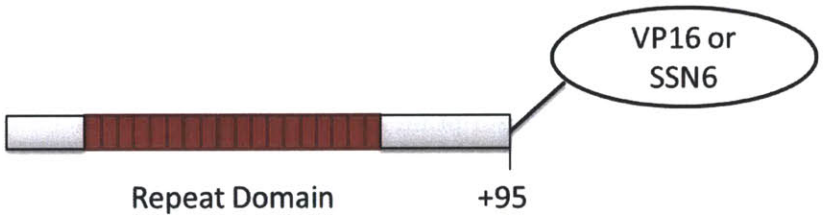
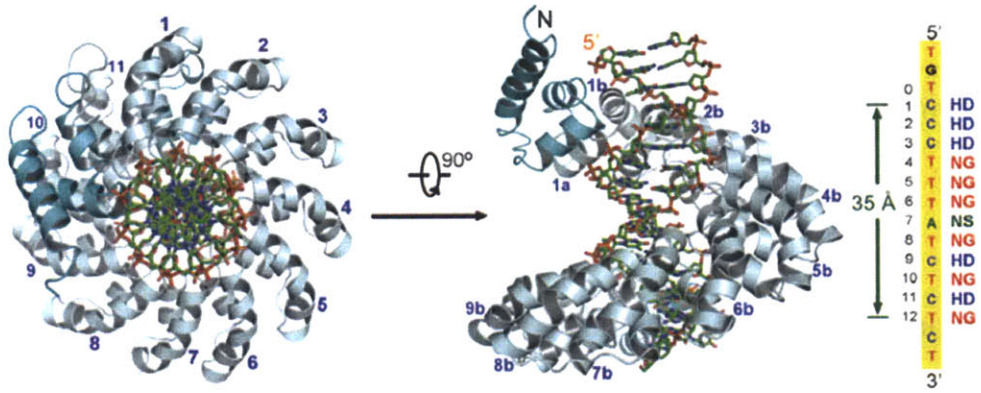


Figure 2: An example of a TALE +95 variant fused to another protein. The fused protein can range from activators like VP16 to nucleases such as FokI. The specificity of the transcription factor binding allows targeting and editing in the genome.

While zinc fingers have been around longer and are generally easier to manipulate in the genome compared to TALEs due to its smaller size, ZFs are harder to program. Zinc fingers have a simple fold that includes an alpha helix, which is exposed in such a way that it can bind the major groove of DNA. The sequence of the alpha helix therefore directs DNA sequence specificity. Only through systematic studies of amino acid replacements in zinc finger proteins have we been able to construct of libraries with known DNA specificities. It is difficult to find a zinc finger that binds to the exact DNA sequence that you want. This is major advantage of using TALEs.

TALEs are secreted by the *Xanthomonas* bacteria, which in its natural state, can bind to promoter sequences in a host plant and activate expression of the plant genes that can aid bacterial infection. TALEs contain multiple, high conserved repeated copies of a 33-35 amino acid repeat

unit and have recently been shown to recognize their target sites by a 1:1 mapping between the repeat residues and DNA bases [1]. The DNA base preference is determined by the identity of amino acids in position 12 and 13 of the repeat unit referred to as the "repeat variable di-residue" (RVD). This simple code between amino acids in TALEs and DNA bases in the target sites suggest that effectively creating TALEs can lead to the ability to bind to and modify any specific gene in the genome. Linking proteins to the TALE can theoretically allow very specific genomic editing [9, 27]. The structure and base preference of the RVDs in TALES can be seen in figure 3.



Weak RVDs – NG (T) and NI (A) (Vander Waals Interaction)
Strong RVDs – HD (C) and NN (G/A) (H-bonding)

RVD:	NI	HD	NN	NG
Base:	A	C	G/A	T

Figure 3: The crystal structures of the TAL effectors as well as a table of the base preference given the RVD. Depiction of the crystalline structure was taken from Dong *et al.* (2012).

In 2011, Miller *et al.* demonstrated the effective application of linking nucleases to designed TALEs linked in order to modify the genome. The FokI nuclease that was linked to the TALE

efficiently generated discrete edits and small deletions within endogenous human NTF3 and CCR5 genes [2]. Those results have spurred efforts to create more TALEs linked to different proteins, not just nucleases, in an effort to better understand how these constructs work.

Historically, the construction of TALEs has been difficult due to the large number of repeat domains. The repeat regions also make amplification of TALE regions with PCR troublesome due to the number of overlaps. Two groups have recently published methods of high-throughput construction of TALEs, one using a hierarchical ligation-based strategy [5], and the other using the FLASH assembly method [3]. However, the specificity and insight into how binding differs as the linked proteins change are still not completely understood and is an area worth studying. Cleaving the C-terminus at various points can also change the TALE activity binding activity.

TALEs are an important building block in the future of synthetic biology. Its robustness, DNA recognition, and the ability to have other proteins fused to it allow it to operate as a synthetic transcription factor, which makes it useful for higher order complex biological circuits. However, one problem with biological circuits and gene manipulation is off target effects. Therefore, modeling specificity and characterizing TALEs better are vital steps before the field can progress. Specificity is crucially important in designing orthogonal biological circuits. We want the circuits to not (or minimally) interact with the host in which they are introduced in order to avoid unintended effects. Orthogonality is useful because off target effects can limit the utility of TALEs for specifically controlling the expression of a targeted gene, which would restrict application of TALEs as components of synthetic circuits where orthogonality is an

important constraint. Work has already been done for characterizing specificity and orthogonality of certain zinc fingers [23].

Identifying exactly how TALEs bind and what factors contribute to binding can help give insight into how genetic variation affects gene expression and structural basis for the TALE – DNA binding. New technological advances have greatly increased speed at which comprehensive binding information can be provided [4]. A predictive model can then be developed for the specificity of TALE binding and activity of attached proteins. This predictive model of how different genomes are affected differently can provide valuable information in various fields such as testing in drug development for human disease treatment.

Methods and Results

My project tests specificity and orthogonality of TALE binding with a two-prong approach. The first approach characterizes the specificity of TALE binding in a methodical way by crossing a library of TALEs fused with either activation or repression domains with a library of reporter strains which contains fluorescent proteins. The binding sites on the reporters will vary and an orthogonality table will be generated based on fluorescent protein expression. From this data we can quantitatively model specificity. The second approach takes a more "shotgun" approach. The TALE will be designed to either bind to the yeast genome or not bind by a certain hamming distance measure in order to characterize the off target binding effects on the genome. ChIP-Seq will then be used to quantify genome-wide effects. ChIP-Seq combines chromatin immunoprecipitation (ChIP) with massively parallel DNA sequencing to identify the binding

sites of DNA binding proteins. It is used to map the global binding sites precisely for any protein of interest, which in this case will be the TALEs [7].

Preliminary work

For preliminary work we tested three TALEs from Addgene: TAL2406, TAL2408, and TAL2409. The TALEs from Addgene are fused to FokI nucleases so they are technically TALENs. We built a new skeletal plasmid that was compatible with the Addgene constructs so that the TALs could be swapped from Addgene to our plasmid through the process of digestion and ligation.

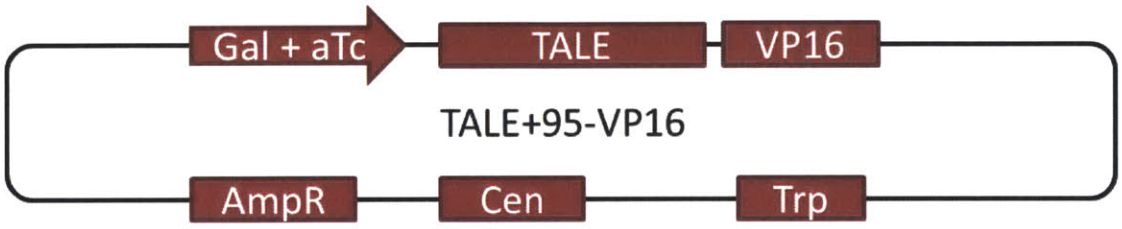


Figure 4: A plasmid map for TALE construct to depict important features. The construct has a Gal + aTc inducible promoter that drives production of the TALE and a fused VP16 activating domain. The construct is also Amp resistant and Trp- for growth and centromeric for transformations.

The new backbone contained a galactose (Gal) and anhydrous tetracycline (aTc) inducible promoter. The TALE segment was compatible with the ones from Addgene and had a +95 amino acid truncated C terminus fused to a VP16 activation domain. We chose to make the plasmids centromeric for ease of integration and replication. We also built reporter plasmids which contained the binding site for one of TAL2406, TAL2408, and TAL2409 fused to green fluorescent protein (GFP). The resulting fluorescence levels after the TALEs and reporter were double transformed into the W303 yeast strain and can be seen in figure 5.

Reporter \ TALE	2406	2408	2409
2406	23040	10657	12973
2408	10737	26806	13687
2409	14657	12484	28564

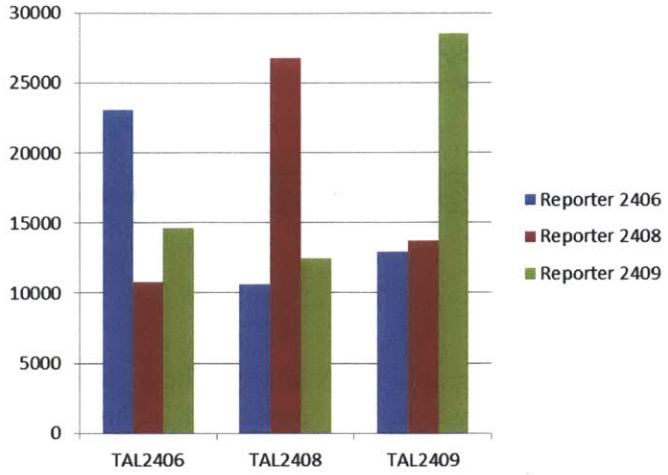


Figure 5: The orthogonality matrix of TALE2406, TALE2407, and TALE2409 crossed with the reporter constructs with those binding sites. All three TALEs showed higher activation and expression of the GFP on the reporter strain when it was correctly matched.

TALE Library Selection

After confirmation that the TALE system worked, we moved forward with our two prong approach of characterizing the TALE-DNA interactions. The first thing that was needed was a library of TALEs and a library of reporters. Since both the crossing with reporters and the ChIP-Seq with the genome required a library of TALEs, the library was selected so that it could be used for both methods. The TALEs were separated into four different categories based on the hamming distance between the target sequences and the yeast genome. Hamming distance was shown to be a good measure of orthogonality by Garg, *et al.* in 2012 [8].

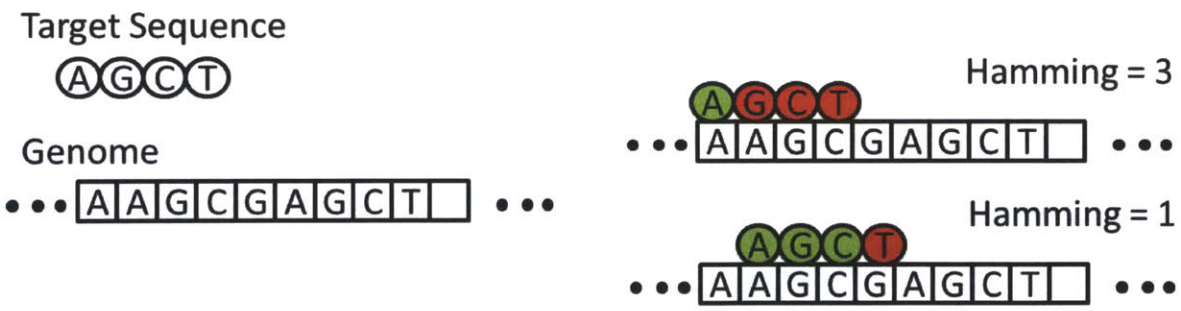


Figure 6: A graphical example of how the hamming distance algorithm works. The number of mistatched base pairs is considered the hamming distance. The number of occurrences for each hamming distance is tracked as the target sequence is moved along the genome.

The hamming algorithm was run on all available TALEs from Addgene to check which ones could be ordered and which ones had to be assembled through the FLASH assembly method [3]. The four categories of TALEs were: (1) hamming distance equal to 1 from the *S.cerevisiae* yeast genome (low hamming), (2) hamming distance between 2 and 3 inclusive from the yeast genome (medium hamming) and (3) hamming distance greater than 3 from the yeast genome (high hamming). The last category had (4) hamming distance equal to 0 that we designed bind to the well-known PHO pathway [12]. The binding sites were chosen to be in the promoter binding region and sites in the open reading frame.

Category	Hamming Dist.	Target	Notes
High Hamming	4	GGCGGGGAGCCGCGGGGA	TAL2351
High Hamming	4	TTGCGCTCGGGGCGGCCA	TAL2340
High Hamming	4	CCCCAGAGCCGTCCGCGA	TAL2269
High Hamming	4	CCGGATCGGGCGCTGCCA	TAL2297
High Hamming	4	TCCCGCAGGGAGCGGACA	TAL2280
Med. Hamming	3	CCGCCGCGCGTACCCCGA	TAL2473
Med. Hamming	2	TTCCTCCCAGGGGGATGT	TAL2205
Med. Hamming	2	CCAACCAAGGAGCATTCA	TAL2369
Med. Hamming	3	CCAGCTCTGGCCCTCAAA	TAL2449
Med. Hamming	2	GCTCAAGATTCAGTAGA	TAL2255
Med. Hamming	3	GCTGGTGGCGTAGGCAA	TAL2273
Low Hamming	1	TGGCATGGCCAGCAACAG	TAL2420
Low Hamming	1	CCAGCACTATTTCTACGA	TAL2281
Low Hamming	1	TATTTCATCCTGCTCTCA	TAL2415
Low Hamming	1	TCGGCCACCATGTCCC	TAL2288
Low Hamming	1	TTTTCAAGTGAAGACAAA	TAL2412
Instances	0	AATTGAATAGGCAATCTC	PHO2 binding site
Instances	0	TATATATTAAATTAGCAC	PHO2 binding site
Instances	0	TGGCACTCACACGTGGGA	PHO4 binding site
Instances	0	TTGGTCACCTTACTTGGC	Random in Promoter
Instances	0	TGAACCCATACACTGGTG	RAP1 binding site
Instances	0	TGATTTGCCTGAAGGTTG	Random in ORF

Table 1: The list of all TALEs that will be used, divided into categories based on the hamming distance

Vector and Reporter Constructs

A new vector was built to satisfy the new criteria for the extended analysis. As before, the vector had to be compatible with the Addgene TALE vectors. However, due to the subset of TALEs that bind to the metabolic pathways, we needed a new promoter system that was inducible in glucose and was orthogonal to yeast. We also built two version of the vector, one containing the activating domain VP16 and the other containing the repression domain SSN6. The new

promoter was chosen to be an aTc inducible cyc1 promoter that was tested to have approximately 200 fold induction.

Since the vector was centromeric, a fluorescent protein was needed to create the appropriate transfer function between expression levels. We could not simply measure concentration of aTc versus the level of reporter expression because the number of plasmid copies in the yeast was variable. To get around this, we fused a GFP to the end of the vector so we can get a transfer function of GFP versus RFP, which was on the reporter strain. In addition, a T2A cleavage segment was introduced before the GFP in order to get greater than 95% cleavage rate [6]. A graphical depiction of the two versions of the new vector can be seen in figure 7. The important features of this new plasmid can be seen in figure 8. It is important to note that all the listed restriction sites are unique so that any individual or combination of components can be substituted. It is built for use by TALEs fused to either VP16 or SSN6 and a T2A GFP but can easily be substituted to be a zinc finger fused to another domain followed by any other fluorescent protein.

The new set of reporter constructs, which contained the TAL2409 binding sites, were built with double binding sites to test the effect that spacing between sites had on binding activity. There were two versions of the reporter constructs: one for activators and one for repressors. The repression reporters had an upstream activation sequence (UAS), a constitutive promoter and red fluorescent protein (RFP). The double binding sites were located between the promoter and the RFP. RFP was chosen to minimize overlap with the wavelengths of the GFP that was on the TALE vector. The activation reporters did not have an UAS sequence and had the TALE

binding sites upstream of the promoter rather than between the promoter and the RFP. The spacer lengths between the binding sites that were tested were 9 base pairs (bp), 11 base pairs, 13 base pairs, 15 base pairs, and 27 base pairs.

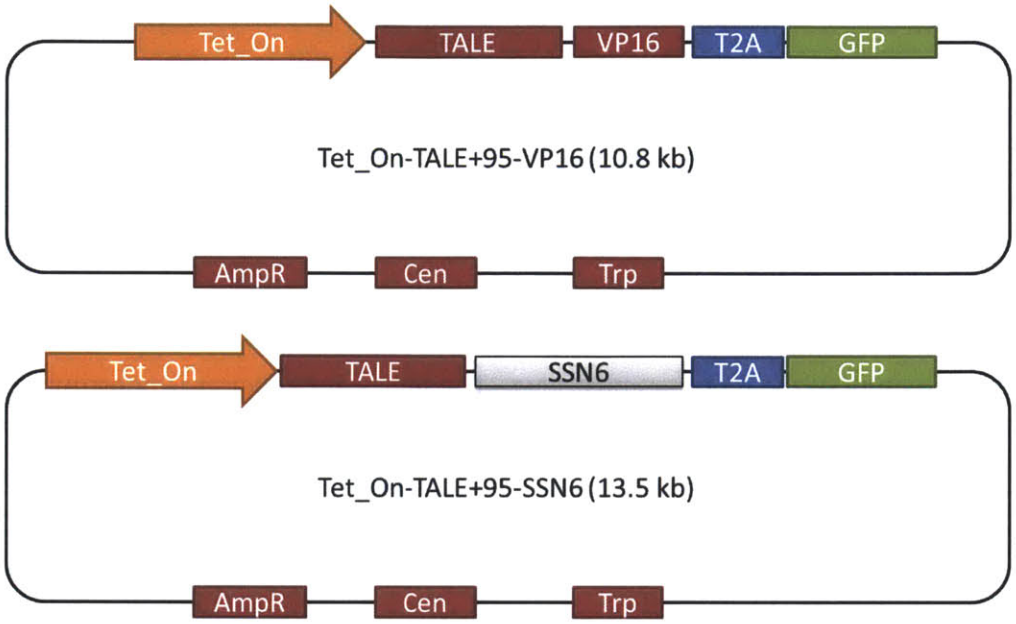


Figure 7: Plasmid maps for the new TALE constructs. There is an activator version, which contains VP16, as well as a repressor version, which contains SSN6. The promoter is now a cyc1 promoter that is aTc inducible and orthogonal to yeast. The fused protein is further attached to a 2A sequence along with GFP so a transfer function can be created.

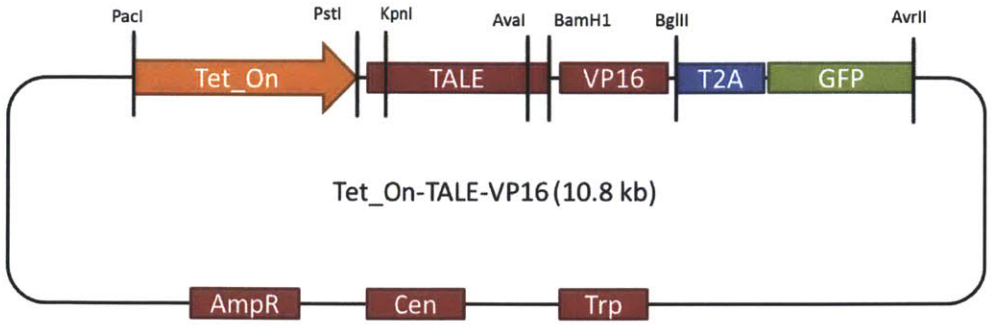


Figure 8: The important restriction sites on the construct. All components can be interchanged.

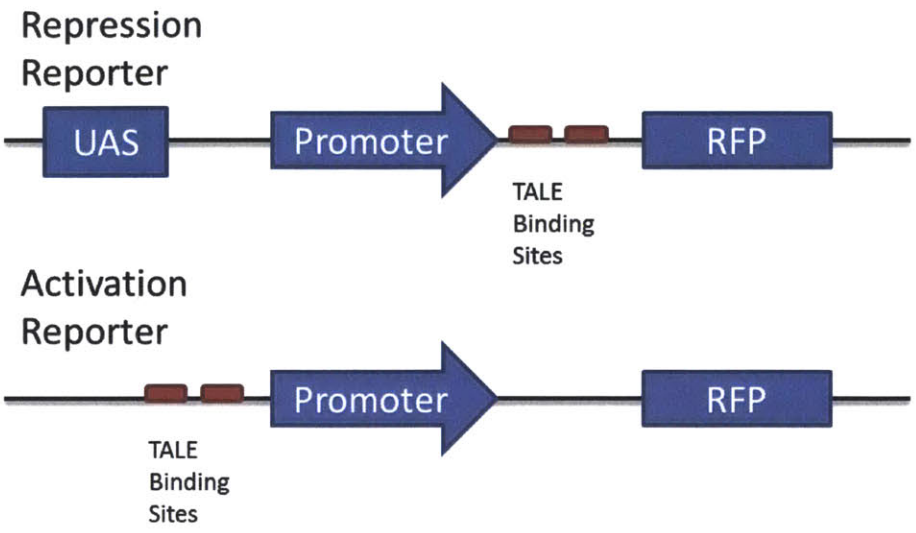
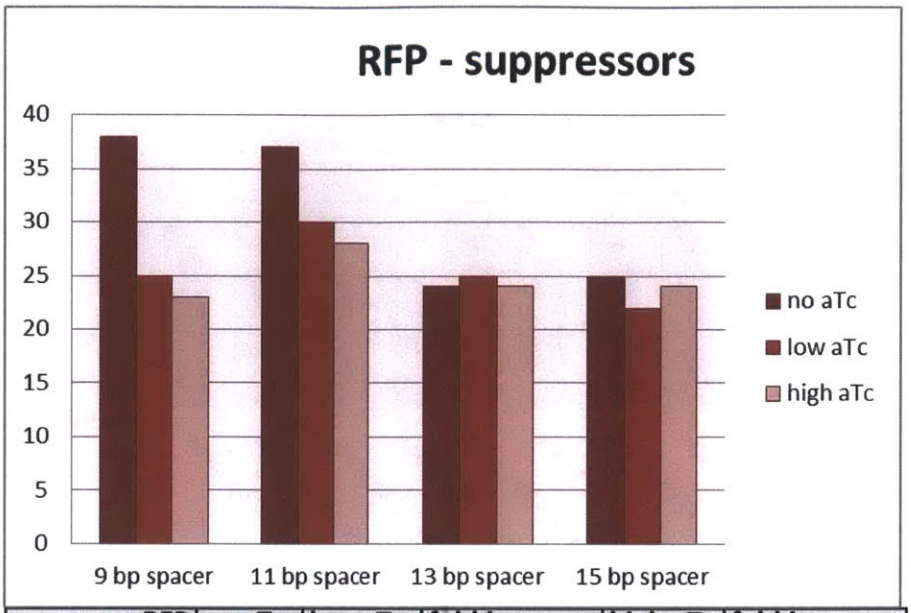


Figure 9: The structure of the reporter constructs. The repression reporter is constitutive for RFP expression with TALE binding sites between the promoter and the RFP. The activation reporter has TALE binding sites acting as the UAS.

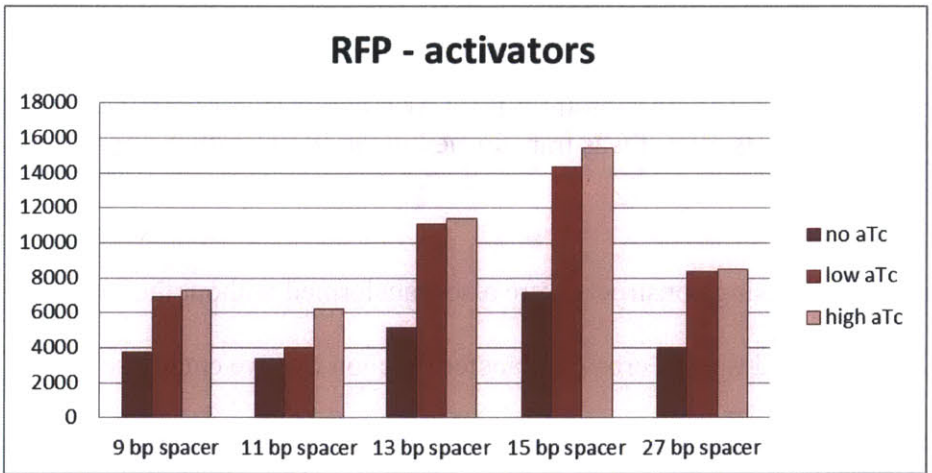
Data

The new vectors from figure 9 containing TAL2409 were transformed into W303 yeast strains that already had the reporter strains integrated in. The constructs containing VP16 were transformed into cells containing the activation reporter and the constructs containing SSN6 were transformed into cells containing the repression reporter. The double transformed cells were then grown to saturation and induced with 0 ng/mL aTc (no aTc), 100 ng/mL (low aTc), and 500 ng/mL (high aTc) for 12 hours. The fluorescent reading for RFP as well as GFP levels were obtained using the LSRII flow cytometer (BD Biosciences). All the data were gated by forward and side scatter and each data consists of at least 10,000 cells. The output of the LSRII can be seen in figures 10 – 12.



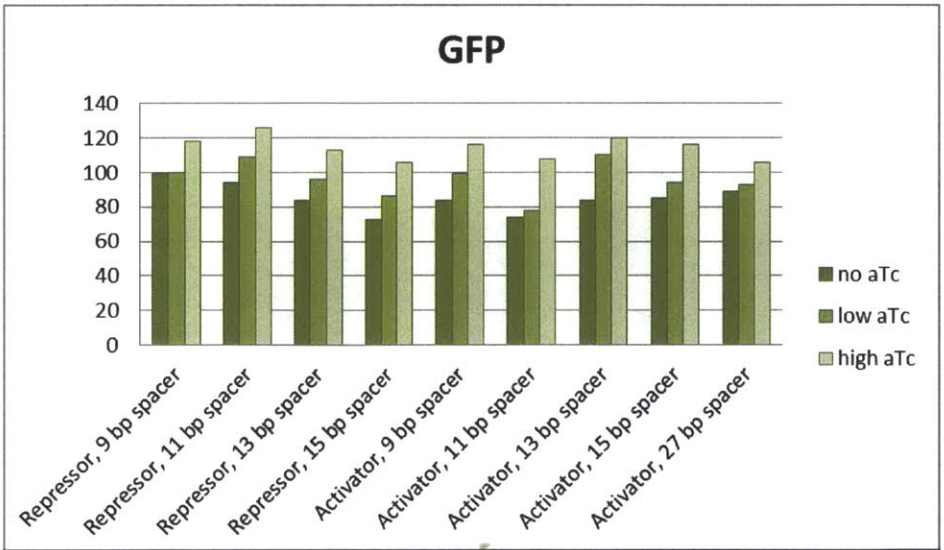
RFP	no aTc	low aTc	fold increase	high aTc	fold increase
9 bp spacer	38	25	0.66	23	0.61
11 bp spacer	37	30	0.81	28	0.76
13 bp spacer	24	25	1.04	24	1.00
15 bp spacer	25	22	0.88	24	0.96

Figure 10: The expression of RFP when the repression reporters are combined with the activating TALE constructs and induced



RFP	no aTc	low aTc	fold increase	high aTc	fold increase
11 bp spacer	3355	4003	1.19	6236	1.86
13 bp spacer	5125	11055	2.16	11353	2.22
15 bp spacer	7135	14328	2.01	15423	2.16
27 bp spacer	3974	8335	2.10	8467	2.13

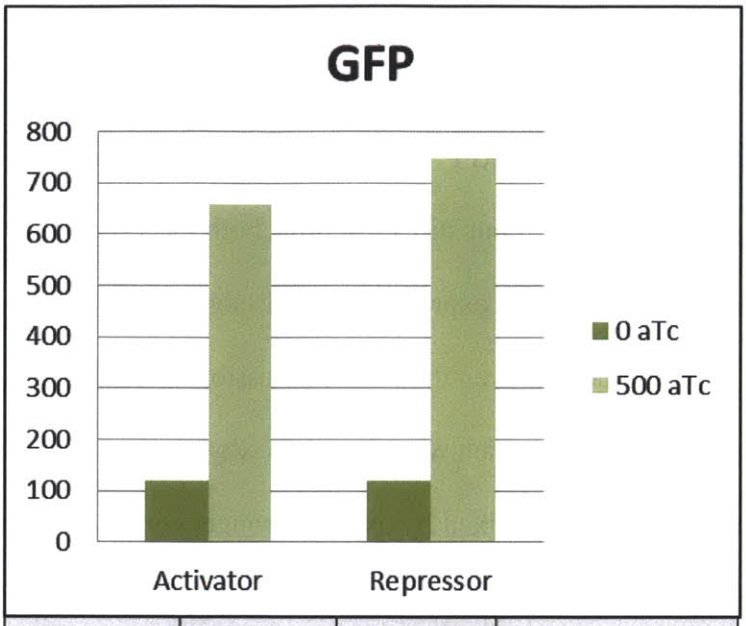
Figure 11: The expression of RFP when the activation reporters are combined with the activating TALE constructs and induced.



GFP	no aTc	low aTc	fold increase	high aTc	fold increase
Repressor, 9 bp spacer	99	100	1.01	118	1.19
Repressor, 11 bp spacer	94	109	1.16	126	1.34
Repressor, 13 bp spacer	84	96	1.14	113	1.35
Repressor, 15 bp spacer	73	86	1.18	106	1.45
Activator, 9 bp spacer	84	99	1.18	116	1.38
Activator, 11 bp spacer	74	78	1.05	108	1.46
Activator, 13 bp spacer	84	110	1.31	120	1.43
Activator, 15 bp spacer	85	94	1.11	116	1.36
Activator, 27 bp spacer	89	93	1.04	106	1.19

Figure 12: The level of GFP expression from the TALE construct for both the activating and repression TALE constructs after it was transformed in the W303 with the appropriate reporters also integrated in.

The activating and repressing constructs were also transformed without the reporters and induced at the same aTc levels and time to create a transfer function that we could use to map GFP production levels to RFP production levels. Transforming the repressor or activator constructs by itself in yeast without the reporters give us an additional control to see whether the reporter is affecting GFP expression levels as well as a more accurate transfer function of aTc levels versus GFP production.



GFP	no aTc	high aTc	Fold activation
Activator	119	658	5.5
Repressor	119	749	6.3

Figure 13: The level of GFP expression from the TALE construct for both the activating and repression TALE constructs transformed in W303 strains without reporters constructs.

Analysis and Discussion

Both the activator and repressor constructs worked but not to the extent that was expected. For the repressor constructs, there is a trend downward for RFP production as the construct is induced by more aTc. However, the overall level of expression was too low for the results to be considered significant. Comparing the basal level RFP levels, RFP production is roughly 100 fold lower in the repressor + reporter construct than in the activator + reporter construct. This tells us that our repressing reporter design might be incorrect because the activating reporter seemed to have generated much better results. Another explanation could be that the aTc induced promoter system is not controlling the TALE construct tightly enough. There also seems to be a correlation between shorter spacers and more repression as can be seen in figure 10.

The activating construct worked much better than the repressing constructs. There seems to be a consistent 2 fold activation increase in RFP when the TALE construct is induced. The 2 fold increase is not as high as we expected but still shows that both the TALE and the reporter constructs are working. The basal RFP expression level is significantly higher than that of the repression construct. It could be that even the basal expression level of TALE-SSN6 was too high for the repressor reporter RFP, which would explain why we do not see much change after induction. If this is the case, it can be fixed by using a stronger constitutive promoter in the repressor reporter construct.

However, the data in figure 12 seems to contradict the explanation that the basal expression level of the TALE construct is too high. The TALE-VP16/SSN6 is linked at the end to a T2A cleaving sequence followed by GFP. The GFP levels from the same FACS run are surprisingly low given the RFP readings. The general trend is still seen, similar to the RFP graphs. The GFP levels increase as the induction levels increase but the magnitude in AU, which is around 100, is too low to be classified as a positive GFP expression since the positive control for a GFP expression cell is usually at least one order of magnitude higher.

We tried to isolate the TALE construct to test just aTc vs GFP expression level without any unintended effects of the reporter constructs. The result was roughly a 6 fold activation increase when fully induced as opposed to the less than 2 fold increase seen with the double transformation. Even then, the fully induced construct only had a GFP fluorescence level of approximately 700 AU, which is still significantly lower than the positive control.

There are a few explanations that we would need to explore. The first is that the TALE construct is too large and the promoter is not fully transcribing through the TALE plus fused domain and protein. If it were somehow stopping only after the TALE, we would see a decrease in RFP and GFP for activation but still a lot of repression due to steric hindrance from the TALE binding. The results support this explanation but there have not been any literature to suggest that TALE constructs can behave in this way. Other explanations include the T2A sequence is not fully cleaving or somehow restricting GFP folding so that is why GFP levels are so low and the two constructs, TALE and reporter, could be interfering with each other in a way that we did not anticipate and do not fully understand right now.

To answer some of these questions, we constructed two new constructs: one that eliminates the T2A sequence and the other uses RFP instead of GFP. These construct test if the T2A cleaving sequence or the GFP folding was the problem. The results from these two constructs were similar to those from the original construct. The RFP expression levels from the first variant and GFP expression levels from the second variant both had around 6 fold inductions. This agreed with what we saw from figure 13 but does not explain why the magnitude of the expression is so low. The best way to test if the TALE was the problem would be to substitute it with a zinc finger. This would not be difficult to do given the design of the vector, but it would also require constructing an entirely new reporter construct, which was not possible given the time restraints.

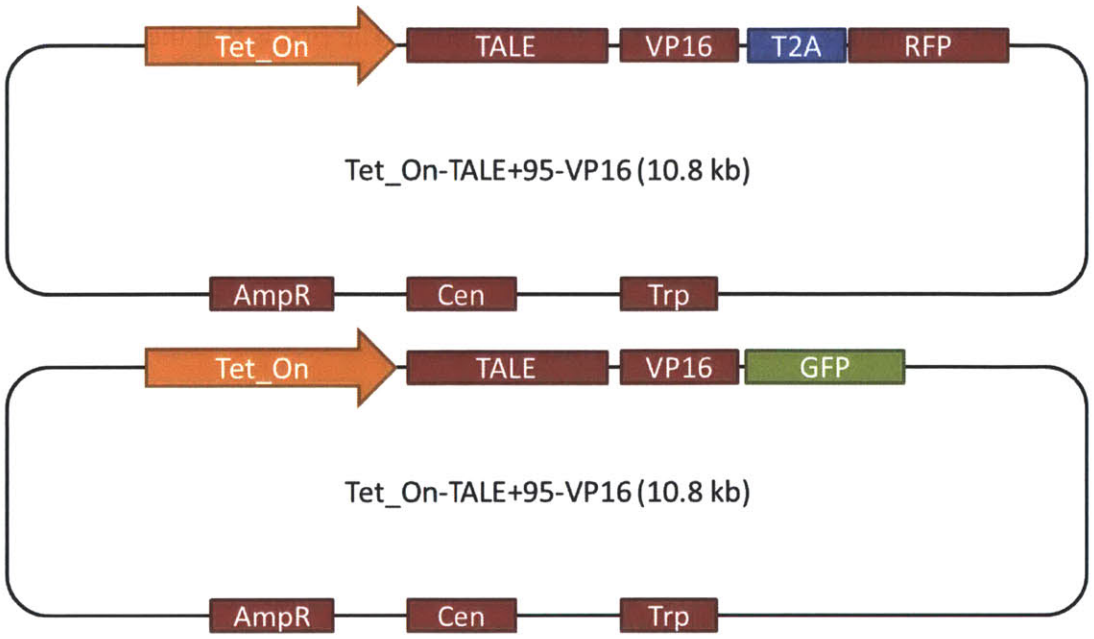


Figure 14: Plasmid maps of the variants of the two variants of the activating TALE construct that was tested. The first variant as RFP instead of GFP; the second variant does not have a T2A sequence.

Conclusion and Future Direction

There is still much more work to be done in this project. The results from this experiment did follow the trend we were expecting but were not entirely conclusive. The zinc finger version of the construct as well as a zinc finger reporter will have to be synthesized in order to verify if using TALEs was the reason for our strange results. After the problems with the constructs are solved, the library of TALEs, those from Addgene and those that need to be synthesized with via the FLASH assembly method, will be substituted in to generate a library of the TALE constructs. A library of reporter constructs will also have be synthesized as well. From there, a series of double transformations will be performed to create a matrix of orthogonality. The library of TALE constructs will also be transformed into yeast and analyzed with ChIP-Seq to see how the

TALE binding affects the general gene expression levels in cell. Depending on the results, we would also want to see how it affects other well-known pathways such as the Gal pathway.

The individual parts of this project can be useful tools for future projects. The hamming algorithm right now is designed so that it can align any desired set of sequences with any desired set of genomes. It is not just limited to TALE binding sites and the *S.cerevisiae*. For example, it can be used to align a list of zinc finger binding sites against the *Pichia pastoris* genome. The activator and repressor constructs were also designed to be as robust as possible. Every single part of the vector can be removed by digesting with a unique subset of restriction enzymes. The new desired piece can then be PCR amplified with the matching restriction enzyme sites and substituted in. The TALE can be substituted for zinc fingers, the VP16 can be substituted for SSN6 or the FokI nuclease to name a few possibilities. This project left a lot of interesting unanswered questions due to time restrictions. However, the components that were generated to solve these problems might end up being more useful than the answers to the problems themselves.

Safe Harbors for Genetic Insertion

Introduction

Escherichia Coli (*E. coli*) has been investigated for over 60 years and is the most studied prokaryotic organism. The bacterium can be easily and inexpensively grown and maintained in the laboratory. Those characteristics make *E. Coli* a model organism for studying genetic manipulation and an important species in both the history and future of synthetic biology.

One of the main methods used to manipulate the genome is integration of synthetic constructs. Once the construct is integrated, the bacteria can amplify the construct when it reproduces and the behavior of the construct can be observed. Ideally, we would choose regions in the genome for insertion of genetic circuits and foreign DNA that are minimally disruptive to the surrounding genes and the function of the organism as a whole. Conversely, we want to be confident that whatever is inserted into these regions is not affected by the surrounding DNA.

Studies have been done in the past to experimentally determine which genes in the *E. coli* genome are important for growth through chromosomal mutagenesis and genetic footprinting [13, 15]. A few integrations sites that would work have been found through these methods. However, these approaches are time consuming and the technique is not robust enough to scale to other more complex organisms. The sites found are also ones that would work but not necessarily the most optimal ones.

We have developed an algorithm to score potential integration sites in *E. coli* in order to systematically generate a list of regions in the genome for integration that would be minimally disruptive. The advantage of the algorithm is that once the technique is experimentally verified,

it can be used to generate similar lists for any organism as long as the genome has been completely sequenced.

Methods and Results

Database and Data Processing

We first find all the transcriptional units in the genome. A transcriptional unit (TU)is defined as a gene or operon plus the associated promoters, transcription factor (TF) binding sites and terminators. Therefore, all genes should be in at least one transcriptional unit. Our primary resource was the *E. coli* database from EcoCyc. EcoCyc is a bioinformatics database the decribes the genome and the biochemical machinery of *E. coli* K-12 MG1655 [14]. It uses a literature-based curation approach and is considered one of the most well-known and accurate databases available. From EcoCyc, we used the Transunits.dat, Promoters.dat, Genes.dat, Terminators.dat, and Dnabindsites.dat datasets. The Transunits.dat contained a list of transcriptional units with all components (genes, promoters, binding sites, terminators) by a unique identifier. It was the umbrella encompassing the other datasets we used. Each transcriptional unit we formed was a structure that had information about the components that formed the unit as well as overall start and ending locations in the genome in terms of base pairs.

After the transcriptional units were formed by combining the components from the appropriate datasets, we pre-processed the data to incorporate a few important assumptions. For every unit, if no upstream binding sites for transcription factors were found, then an 100 base pair buffer was added upstream of the first promoter; if no terminators were assigned, then a 60 base pair

buffer was added downstream to the end of the last gene; and if two or more transcriptional units overlapped, then they will be treated as one single unit.

The TUs also contained information about whether the genes that they are comprised of are essential genes, hypothetical genes, or cryptic genes. Gene essentiality was set to be the genes that were experimentally classified as important by Gerdes *et al.* in 2003 [13]. The EcoCyc Genes.dat dataset contained information about whether a gene was hypothetical or cryptic. If any of the genes in the TU were classified into one of those three categories then the TU was classified into that category. Note that the TU classifications can overlap (e.g. a TU can be both hypothetical and essential).

Criteria for safe insertion regions

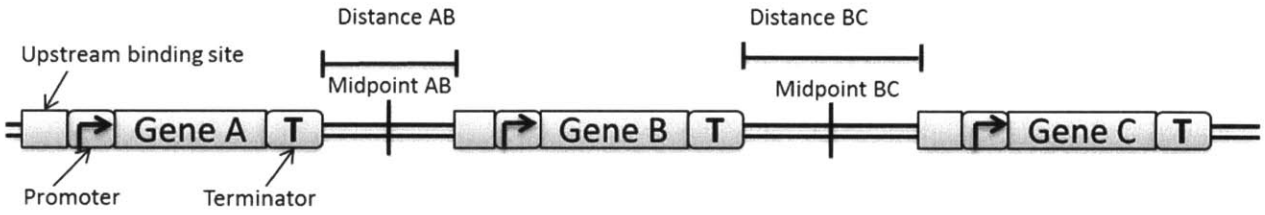
We tried to quantify how good a region is for insertion with the following criteria:

1. It could not be inside any transcriptional region
2. Maximal distance from genes to either side of the integration site
3. Preferably not neighboring essential genes
4. Preferably neighboring hypothetic or cryptic genes
5. Minimal self-similarity between the integration site and the rest of the genome.

Algorithm Overview

The algorithm we designed sorts the transcriptional units based on position in the genome. It then finds the midpoint of the region between successive units. The midpoints will be considered for insertion points. The first score that the midpoints are assigned is based on the

distance between two successive units. An example of the initial scoring can be seen in figure 15. This ensures points 1 and 2 in the list of criteria will be satisfied. From this measure, the top 10 sites can be seen in table 2, and a histogram of the distribution of all distances can be seen in figure 16.



Insertion Point	Distance
Midpoint BC	Distance BC
Midpoint AB	Distance AB

Figure 15: The insertion sites are only chosen from intergenic regions. The site location is the midpoint between the two flanking units and the distance score is the distance between the flanking units.

Top 10 sites by distance	
site (bp)	distance
306377	5742
2330108	4739
1219976	3984
2785404	3890
4573444	3861
2196383	3635
1526541	3610
1470369	3553
249487	2586
1464619	2531

Table 2: The top 10 insertion sites based only on the distance metric.

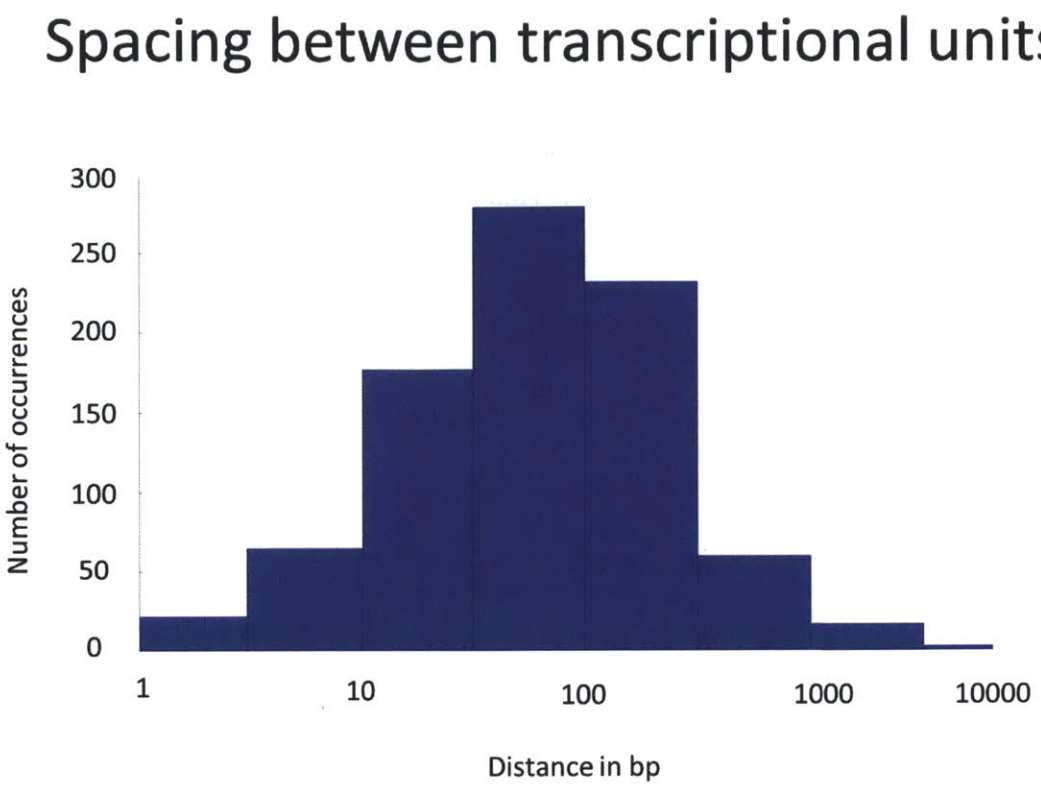


Figure 16: A histogram of all the distances between the transcriptional units found by the algorithm.

In order to account for the fact that we do not want the insertion point to be flanked by essential genes and we would prefer to have the point flanked by hypothetical or cryptic (which are considered less essential), a penalty score is subtracted from the distance score for each flanking TU that is classified as essential. Along the same lines, a preference score is added to the distance score for each flanking TU that is classified as hypothetical or cryptic. The final score will be a linear combination of all the factors, and the best site will be the site with the maximum final score. This addition ensures points 3 and 4 in the criteria will be satisfied.

Self-similarity was quantified by BLASTing the 70 bp homology arm to the left and right of the insertion site against the MG1655 complete genome. A low BLAST score means that there is not much similarity between the sequence and the genome, which is preferable in this application because that is the targeted region for insertion. This feature incorporates point 5 in the criteria list.

We devised a final formula that linearly combines all the above measures to generate a final score. We wanted a scoring function was comprised of distance and essentiality contributing 30% each to the final score, and cryptic, hypothetical, BLAST left and BLAST right all contributing 10% each to the final score. Based on the numbers we were getting from the different measures, we found an approximation to the weighting in the final score as shown below.

Final Score = Distance

+ 5000*(-1*number of essential flanking elements)

+ 1000*(+1*number of hypothetical flanking elements)

+1000*(+1*number of cryptic flanking elements)

+1*(-1*BLAST score left 70 bp)

+1*(-1*BLAST score right 70 bp)

The top 10 final scores and the sites can be seen in table 3. It is interesting to note that none of the top sites by distance were flanked by essential elements; therefore the distance metric remained the highest weighted factor. The final rank is not much different from the rank based solely on distance.

site (bp)	distance	number essential	number hypothetical	number cryptic	blast left	blast right	score	distance rank	final rank
306377	5742	0	0	1	180	318	6244	1	1
2330108	4739	0	1	0	428	530	4781	2	2
2785404	3890	0	0	0	182	212	3496	4	3
1219976	3984	0	0	0	154	396	3434	3	4
4573444	3861	0	0	0	348	290	3223	5	5
2196383	3635	0	0	0	458	154	3023	6	6
1470369	3553	0	0	0	345	261	2947	8	7
1526541	3610	0	0	0	692	621	2297	7	8
249487	2586	0	0	0	708	497	1381	9	9
1464619	2531	0	0	0	586	928	1017	10	10

Table 3: The final top 10 insertion sites found by the algorithm. Since none of the units flanking the insertion site were essential, the distance metric was the dominant determinant, as can be seen by comparing the final rank with the distance rank.

Analysis and Discussion

It will not be possible to comment on how effective this approach to finding integration sites is or whether the weightings in the scoring function are correct until the sites are tested experimentally.

One concern with our approach of algorithmically determining the best site is its dependence on the quality of data. We classified 287 genes as essential based on the Gerdes paper, 15 genes as cryptic, and 123 as hypothetical based on the annotations in the EcoCyc database. There were other databases that had different number of essential, crypt, and hypothetical genes but we chose to classify everything based on EcoCyc because it is known to be well curated and maintained. One example is RegulonDB, which contained more cryptic genes, but EcoCyc is an aggregate of the data from RegulonDB among others and chose to only include a subset of the RegulonDB cryptic genes as cryptic. There were too many small discrepancies between

different databases and we decided to base our algorithm off only one because it was not possible to know which one is most accurate.

Once we get experimental data on how the top integration sites are performing, we can apply a machine learning approach to extract values for the weightings of each different component of the final score.

Conclusion and Future Direction

This project is currently in the experimental validation stage. Constructs have been designed and are currently being integrated into the top sites from table 3. The behavior of the constructs and the *E. coli* in general will be assayed to quantify how much better it is to integrate at one of these top sites. Assuming the integrated circuit behaves as expected, quantitative PCR can be used to verify minimal impact of the circuit on nearby genes and microarrays can be used to verify minimal impact of the circuit on the entire cell.

If this approach to finding safe harbors for insertion passes experimental validation, the algorithm can be extended into other organisms such as yeast and mammalian cells. The approach is independent on organism type and can easily be adapted for other genomes. The only requirement is a well-curated database of how the genetic components are regulated. Sites for eukaryotic cells are significantly harder to predict than prokaryotes due to introns and much more complex system of transcription factors and transcription units.

Genetic Encryption

Introduction

Synthetic biology has emerged as an engineering discipline for biological systems, with a diverse set of applications. It is the application of engineering principles to the realm of biology for the purpose of producing novel and useful biological components. As the field of synthetic biology progresses, the quality and quantity of synthetically built circuits has risen. Circuits ranging from two input Boolean logic functions [16, 22] to genetic programs that integrate environmental sensors and control expression dynamics [20] have already been created, and the future will bring even higher-order circuits with broader ranges in application and complexity.

While creating an array of tunable and characterized parts for use in circuits is still a challenge right now, creating programmable functional synthetic gene circuits is one of the ultimate goals of synthetic biology [26, 27]. One issue that is introduced along with the more complex designs is the protection of the intellectual property behind these designs. For example, a company specializing in metabolic engineering such as microbial production of biofuels or high-value chemicals would be susceptible to their strains being stolen and sequenced, which would allow another party to replicate their work. Furthermore, the company might not want their strain to be able to reproduce or used after a certain period of time, so how can that be prevented?

Another application that has garnered a lot of attention lately is using DNA as a medium for storing information. Goldman, *et al.* created a scalable method of storing information, encoding computer files totaling 739 kilobytes of hard-disk storage into a DNA code. They then synthesized the DNA, sequenced it and reconstructed the original files with 100% accuracy. The encoding method worked by converting the ASCII code into base-3 using a Huffman code that

replaces each byte with five or six base-3 digits (trits). This in turn was converted *in silico* to a DNA code by replacing each trit with one of the three nucleotides different from the previous one used, ensuring no homopolymers were generated [18]. This method made the DNA more compact but at the cost to security. If the encoded information were sensitive, then another layer of protection might be needed in addition to the current encoding. There are other DNA storage methods such as the one developed by Church, *et al.* that has higher level encoding security but requires next generation technology in DNA synthesis and sequencing at the cost of DNA compactness [17]. Even then, adding another layer of security could only be beneficial.

In this project, we propose using the method of overlapping recombinases to encrypt genetic information. We developed a way of encrypting the topology of a synthetic gene network, as well as a way to decrypt it again to recover the functional network. The idea is that the network would be stored *in vivo* in the secure encrypted form, and only decrypted when needed. This method is robust enough to be an added layer of security, independent of the nature of the application. Unlike the encoding schemes previously mentioned, the overlapping recombinase method can be used whether it is scrambling a genetic network, or adding a layer of encryption onto the encoded storage DNA. A flowchart for the process and reasons for our method can be seen in figure 17.

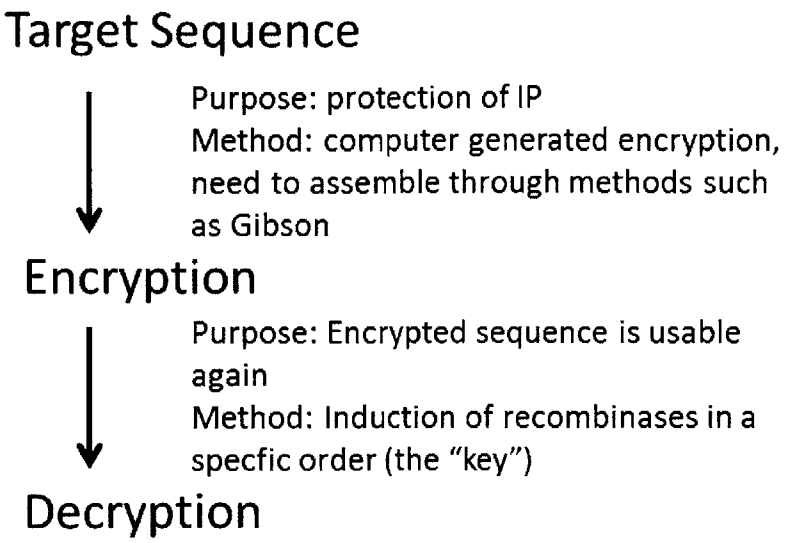


Figure 17: The flowchart and general purpose of the algorithm. The encryption is *in silico* and the encrypted sequence will need to be assembled *in vitro*.

Recombinases are enzymes derived from bacteria and fungi that catalyze directionally sensitive DNA exchange reactions between short (30-40 nt) target site sequences that are specific to each recombinase [19]. They are widely used in multicellular organisms to manipulate the structure of genomes and control gene expression. The natural role of recombinases was not control gene expression but rather to insert phage DNA into host genomes. Recombinases recognize a pair of specific nonidentical sequences known as attB and attP. After recombination the sites are converted into attL and attR, respectively, which no longer serve as recombinase substrates [19-22]. The main idea behind controlling gene expression is flanking important components with attP and attB sites so that the recombinases are "tricked" into manipulating the flanked sequences. The unidirectional recombinases, such as Bxb1 and phiC31, can irreversibly invert or excise DNA on the basis of the orientation of the surrounding pair of recognition sites (figure 18) [19-22].

Currently recombinases have mainly been used to build a variety of single layer digital logic architecture. Single layer is ideal because it allows truly digital logic as well as simpler networks that produce the same function as cascades. Without relying on cascades, the recombinase controlled gene expression can exist only in a finite number of states. In terms of logic gates, it would be either off or on without the intermediate states. Siuti *et al.* used unidirectional recombinases to build all 16 two input Boolean logic functions by flipping gene promoters, transcriptional terminators, and gene-coding gene-coding sequences [22].

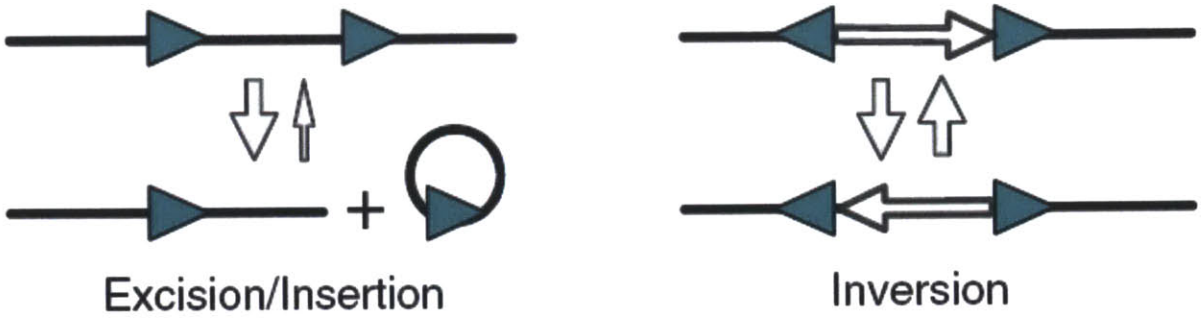


Figure 18: Recombinases with attP and attB sites facing in the same direction will excise the segment the sites are flanking. If the sites are facing opposite directions then the flanked segment will be reverse complemented.

One of the most widely used encryption methods is on the RSA algorithm. RSA is a public-key cryptography system that is based on the factoring problem, or the difficulty of factoring large integers that are the product of large prime numbers. The integer factoring problem is known to be in the non-polynomial deterministic time (NP) complexity class, which means no current algorithm that has been published can factor these large integers in polynomial time [28]. The overlap recombinase encryption method applies some of the characteristics of RSA cryptography to the biological world by making the combination of possible outcomes during decryption so

large that it will be impossible to know which decrypted sequence is the correct one without knowing the decryption "key".

Methods and Results

Encryption

The main idea behind the encryption method is to use overlapping recombinases to shuffle, or `scramble' the promoter-CDS (coding sequence) pairings that defines the topology of a gene network. `Decoys', which can be either unused recombinase sites, or genes that are not necessary for the network to function, can be added to increase the level of encryption (make it harder to decrypt). The innovative features that make encryption possible are: (1) the overlaps allow the flanked segments to get shuffled (either out of order or back into order) and (2) the order of the shuffling generates a "key", which is the order that the recombinases need to be induced in so that the final product is correct. If the recombinases are induced in a different order than the one specified, the final product will be incorrect and can be drastically different from the target product.

The algorithm we developed allows you to specify how many recombinases you want for shuffling steps and how many decoy recombinases or genes you want to insert, as well as the makeup of the decoy components. Currently, the algorithm is stochastic and randomly selects where to place the recombinases and decoys based on a few simple heuristics. A simple example of how the encryption works can be seen in figure 19.

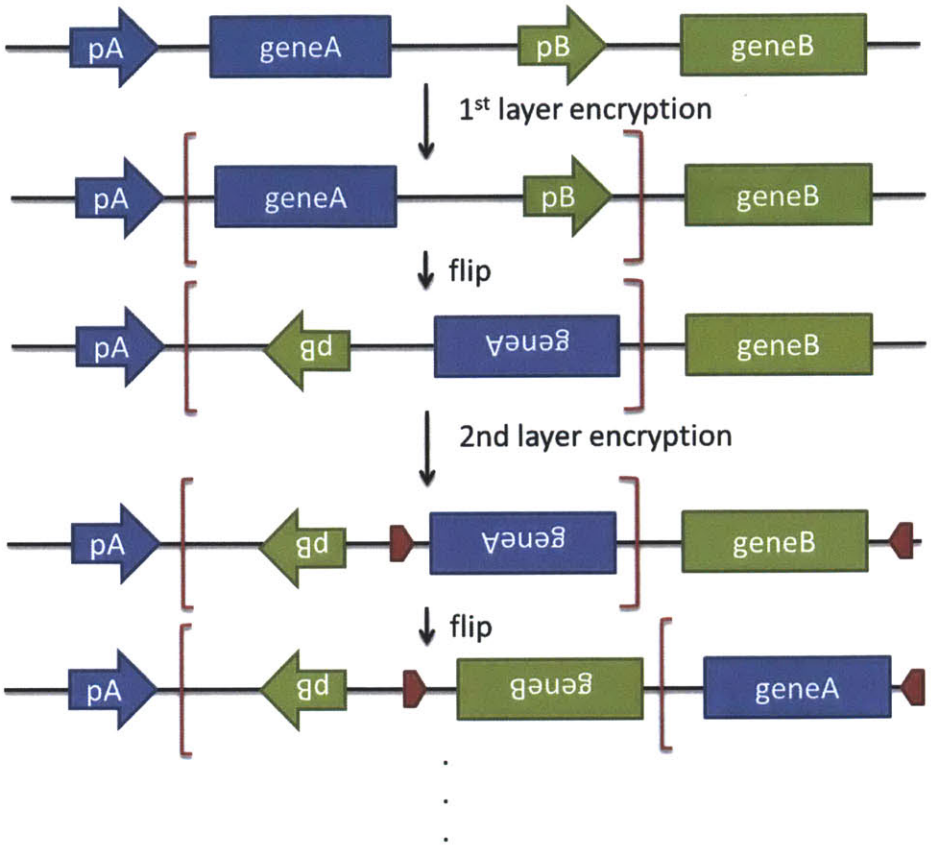


Figure 19: An example of how the shuffling in the encryption works. The recombinases are randomly placed (can be chosen by hand if preferred). The flanked segment is then reverse complemented. The additional layers will repeat this process but have overlapping flanked segments so that the order of the components gets scrambled. This method ensures that we can get back to the original target sequence during the decryption process.

The amount that the DNA gets scrambled depends on the number of recombinases used, which corresponds to the layers of encryption, as well as how well the placement of the recombinases and overlaps are chosen. When decoy components are added, they are flanked by a recombinase halves that face the same direction. This will allow the decoy component to be excised when the flanking recombinases are induced as explained in figure 18. An example of how decoys work can be seen in figure 20. The ability to excise can produce drastically different final products if

multiple layers are used and the recombinases are not decrypted in the correct order as seen in figure 22.

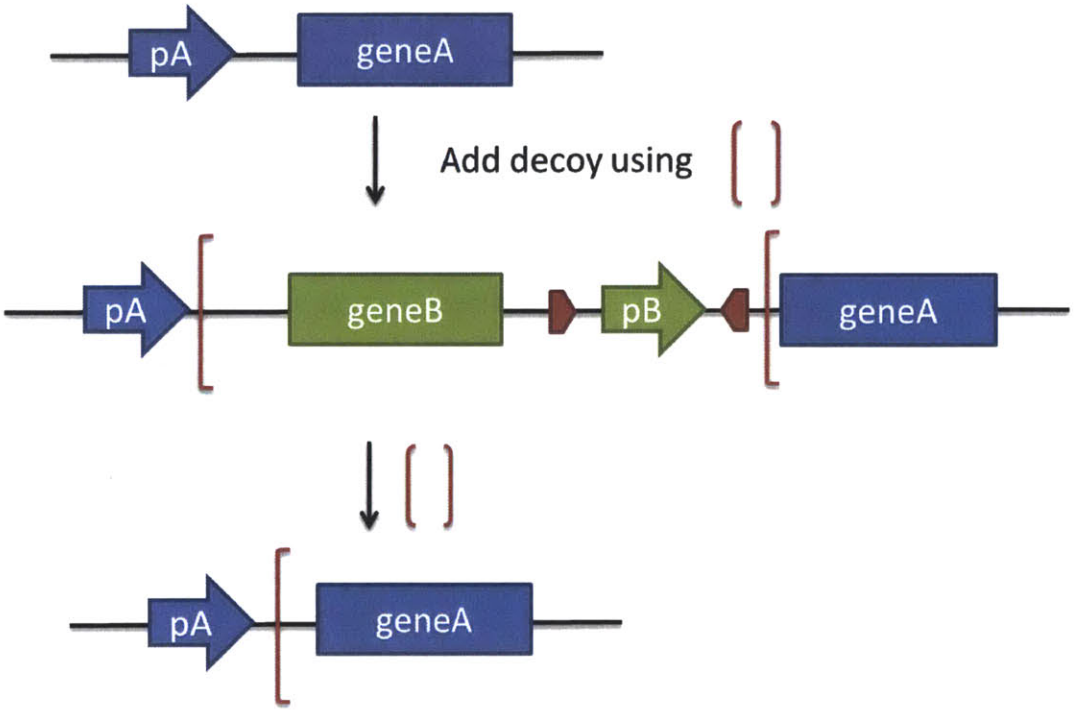


Figure 20: An example of how adding in decoy in the encryption algorithm works. A random site is chosen and a recombinase with sites facing in the same direction is added. The flanked segment will be the decoy and can contain any amount of decoy components. Note that during decryption, if the recombinase is induced the entire decoy segment would be removed.

As the number of layers of encryption and the number of decoys added increases, the original sequence becomes increasingly scrambled. Some additional features are that in decoys, we can add in not only outside decoy components (e.g. a new gene or promoter) but also previously used recombinases (recombinases that were used to encrypt layers below the current one) and repeats of any exisiting components. Adding in previously used recombinases in the decoy would introduce more sets of the same recombinase site; but if the encryption is decoded in the correct order, then the repeat sites would be removed when following the decryption steps, which is why it is important recombinases used in future encryption steps are not used in the decoy at a given

step. Theoretically the additional repeated recombination sites would generate even more valid decryption possibilities, but if they are not added carefully, then it can used to track relative ordering of the recombination steps. In addition to the decoy components and extraneous recombinases, it is important that only previously used recombinases (a feature that can be left out if desired) are used in the decoys otherwise the decryption process will not work properly.

Decryption

The encryption works in layers by using a new recombinase at every step. The only correct way to decrypt the encrypted sequence would be the reverse order of the encryption steps as shown in figure 21. If different steps are followed then you can get vastly different results as seen in figure 22.

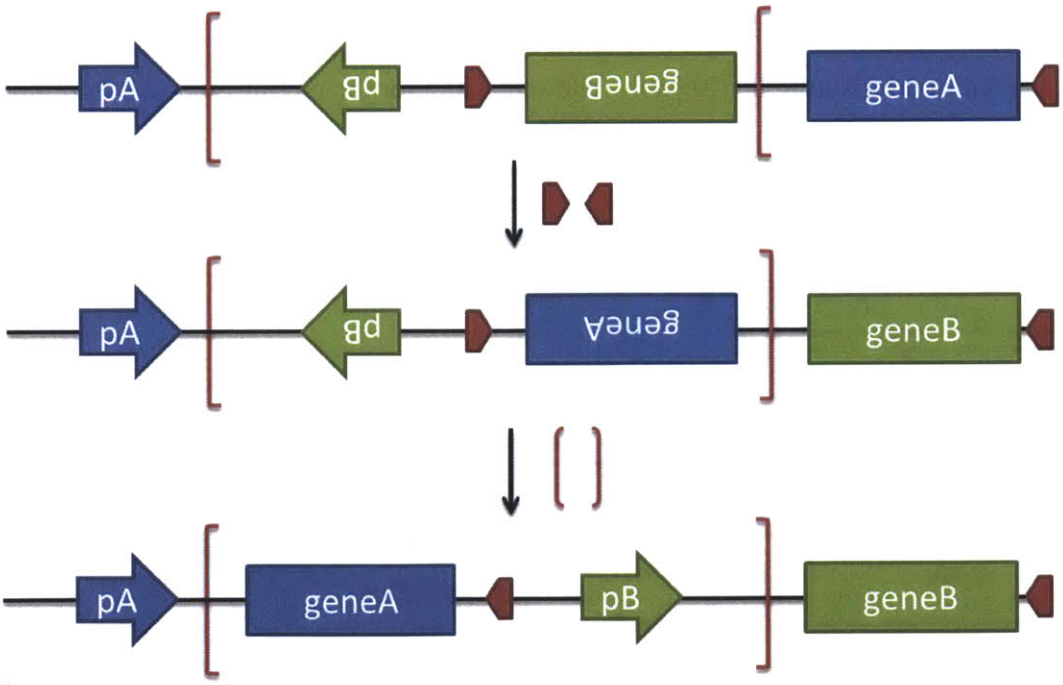


Figure 21: An example of how the decryption process works. The encrypted sequence is the output of figure 19. If we induce in the opposite order of recombinases that the sequence was encrypted in, then we will get the target sequence back again.

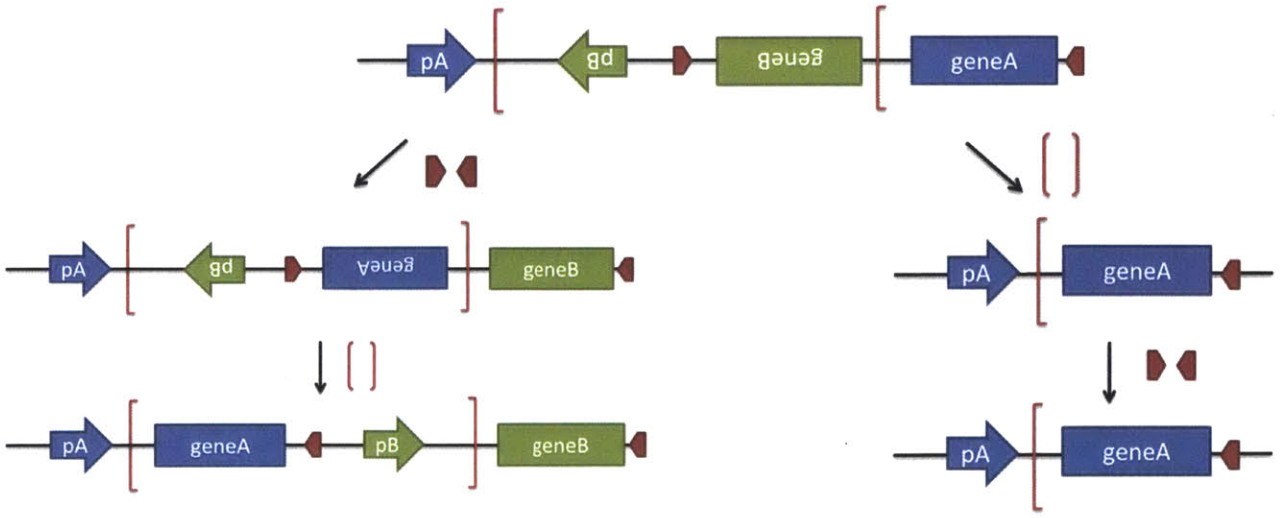


Figure 22: The two possible decryptions from the encrypted final product in figure 19. If the order of induction is incorrect the final outcome can be significantly different than the actual correct sequence.

Applied Example

As an example of encrypting a real synthetic gene network, we encrypt the 4-input AND gate construct from the Voigt lab (Nature 2012) [20]. The schematic for the circuit is shown in figure 23 and the layout of the construct is shown in figure 24.

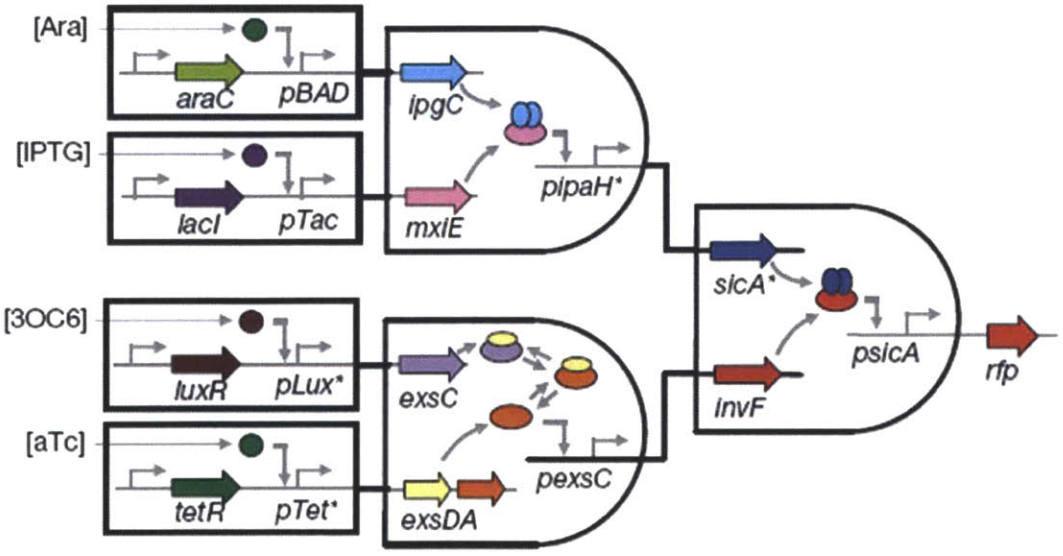


Figure 23: Circuit layout of the Voigt circuit. Picture taken from Moon *et al.* [20]

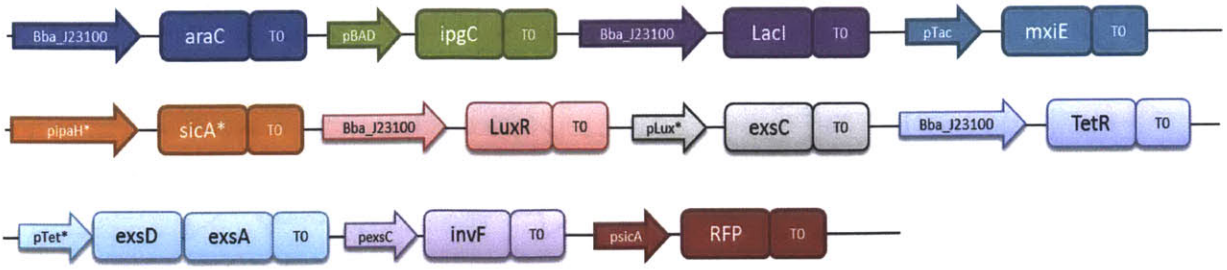


Figure 24: The reconstructed Voigt sequence as it would appear on a plasmid. Since this was a proof of concept example, we did not include features such as ribosome binding sites.

We replicated the format of the construct as best we could from the supplementary material in the online version of the paper. We chose to keep the all ORFs and the associated T0 terminators together and not scramble them. For simplicity we have also not included Ribosome binding sequences. Two new Promoter-TF pairs were introduced as decoys. The first pairng was the lambdaPR promoter and the lambdaCI TF. It was used in the original represillator paper. The second combination was the glnAO promoter and the glnG TF, which were used in a version of the toggle switch. We also limited the number of recombinases to 4 for shuffling and 2 for decoys, which is plenty for the purposes of this example. This example is meant to demonstrate the feasibility of the approach and put it in a physical setting, rather than be a usable example of encryption. Using the encryption method but optimized for this specific application, we came up with the encrypted version of the construct shown in figure 25.

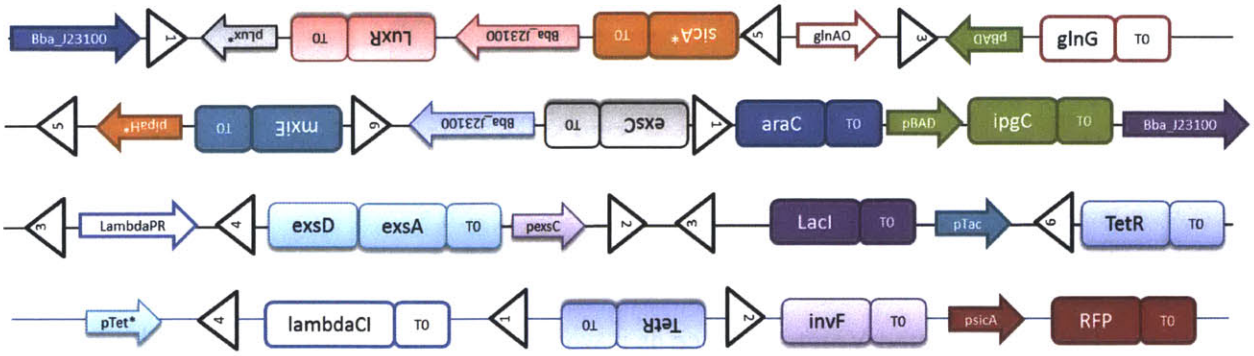


Figure 25: The encrypted output of our encryption method on the Voigt circuit.

The encryption steps were conveniently ordered as recombinases 1, 2, 3, 4, 5, 6. Recombinases 2 and 5 were used for insertion of decoys and the rest were used for shuffling. To decrypt this encrypted sequence, you would need to induce in the reverse order, or in this particular case, recombinases 6, 5, 4, 3, 2, 1. The final decrypted product can be seen in figure 26.

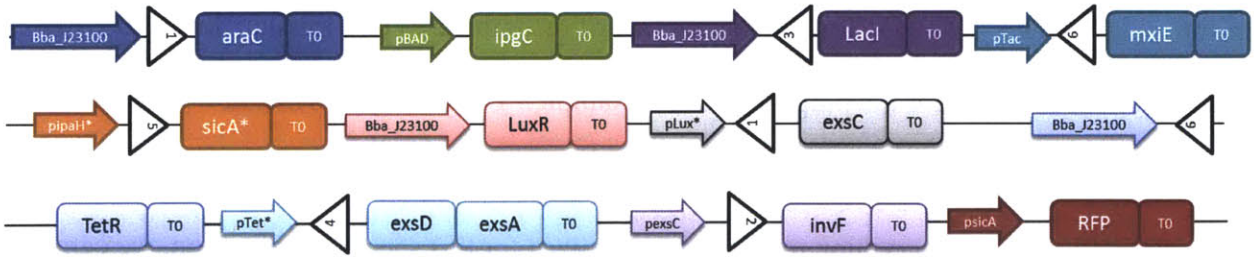


Figure 26: The decrypted output if the encrypted output was induced with the correct ordering of recombinases or "key". Note that the ordering is the same as the original sequence in figure 24.

Analysis and Discussion

In the examples provided in this document, the goal was to separate the gene from its associated promoter to change the network topology. The encryption of the Voigt network would probably be relatively simple to crack given the low number of recombinases used. However with a sufficient number of recombinases and decoys, the number of possibilities becomes prohibitively

large. However, the algorithm does need to be optimized to fit some applications that are too sensitive to just require stochastic placement of the attP and attB sites.

Computational Complexity

Suppose N is the number of recombinases used. If we ignore the additional complexity introduced by decoys, with each additional layer of encryption and recombinase used, the number of possible decryptions grows by O(N), while the encryption step requires O(1) computations. From this we see that the encryption is a O(1)*N = O(N) operation and the decryption is 1*2*3*...*N=O(N!) operation if you do not know the key. If you do know the key, then decryption is also just O(N). When decoys are taken into account, encryption is still a O(N) operation; but the complexity of decryption increases asymptotically faster than the complexity without decoys. The scaling of complexity for the no decoys case can be seen in figure 27.

This is an ideal type of growth for an encryption system. It is easy computationally to generate more difficult encryptions but the encryptions get significantly harder to crack. Two important things to note are (1) if most decrypted versions are valid circuits then you would not even know if you came across the right answer and (2) O(N!) is in the NP complexity space, which means that in the future if we get more recombinases and N gets big enough, the number of different permutations would not be able to be listed even by computer [28].

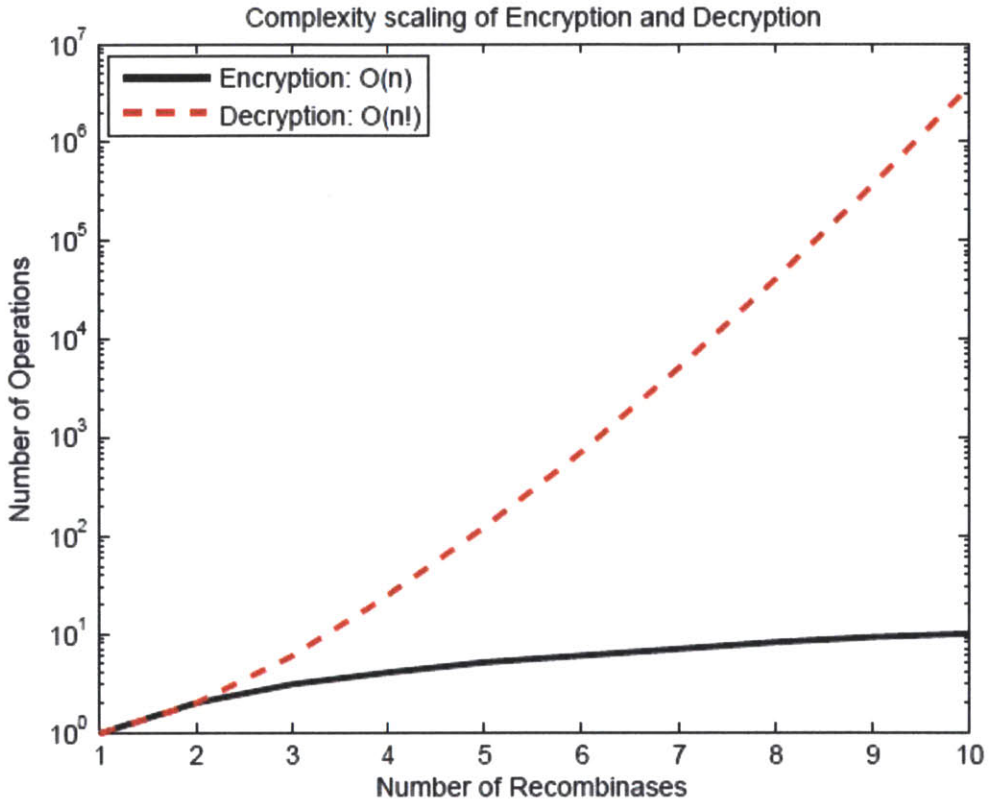


Figure 27: The complexity growth of encryption and decryption for O(N) and O(N!) as N increases. O(N!) is significantly more complex asympototically. Even for N as low as 10, O(N!) becomes more than 6 orders of magnitude larger.

Cracking the Encryption for Network Topology

For encryption without any decoys used, if we use N overlapping recombinases then there are N! possible decryptions. For example, for 10 recombinases , there are 10! = 3,628,800 possible ways of decrypting the sequence. Therefore enumerating through all decryption routes is very difficult. Furthermore, many of the solutions will be valid network topologies, so differentiating between the correct solution and all the incorrect ones is not simple.

The fastest method for cracking the sequence is to first sequence the DNA, identify the promoters and coding regions, and then construct all possible promoter-coding region pairings. For a 10 gene network there are 10*10 possible pairings, which is only 100, and one of these would be the correct network. It would be much quicker to enumerate thorough these than the 3,628,800 networks formed by decryption. However, each of these 100 possible networks is arguably an equally valid network topology, so it would again not be simple to identify the `correct' network from the `incorrect' networks.

The addition of decoys can increase the level of encryption substantially. This is because decoys add uncertainty. For instance, for a 10 gene network, which uses decoys genes in the encryption, it is unknown to the third party trying to crack the sequence how many genes are decoys. For sake of argument if (after sequencing and identifying the promoters and genes) the third party thinks that at most 4 decoys could reasonably have be used, then there are $\binom{10}{4}$ subsets + $\binom{10}{3}$ subsets + $\binom{10}{2}$ subsets + $\binom{10}{1}$ subsets + $\binom{10}{0}$ subsets (i.e. if no decoys were used), different options for which decoys could be used. This is 210 + 120 + 45 + 10 + 1 = 386 different possibilities (if they are correct that no more than 4 decoys were used). Because of this the number of possible decryption routes grows much larger (as each version with a subset of the decoys removed has itself N! possible decryption routes). More importantly the number of possible promoter-gene pairing also increases, as each 386 possible versions has N*N possible gene-promoter pairings. In this case the number is $[210*(6^2)]+[120*(7^2)]+[45*(8^2)]+[10*(9^2)]+[1*(10^2)]$ = 17230 possible gene-promoter pairings, which is a substantial increase on the 100 possibilities when decoys are not a potential part of the

encryption. Also using decoy recombinase sites will further increase the uncertainty and level of encryption in much the same way.

Conclusion and Future Direction

There is still a lot more to be done before this method is perfected. If this method works experimentally, we believe the implications could be huge. Protection of information is an important field and any resources that can be used to achieve this would be very useful. We believe that this project has the potential to be high impact in the future due to the simplicity and ease of encrypting and decrypting combined with the difficulty of cracking an encryption. There are a few important practical considerations when implementing the encryptions examples above:

1. The recombinases used must be highly orthogonal. We currently have around 10 – 12 highly orthogonal recombinases that were tested experimentally and more that have not been verified yet
2. The expression of the recombinases must be under very tight control. The induction percentage needs to be around 100% otherwise the population will not be homogeneous and the decryption would result in many different final products.
3. At the moment, the recombinase attB and attP sites must be placed between the promoter and the genes for a network topology encryption. Based on current results, we do not have any reason to believe the addition of recombinase site between the promoter and ORF will adversely affect gene expression.

One of main strengths of this method is how robust it is. The only criteria to what sort of things can be encrypted is the ability to be separated into different modules. The current algorithm can be modified in a number of ways. You can specify the number of components, decoy components, decoy recombinases, shuffling recombinases. It works in a stochastic manner such that where which steps are additions of decoys and which are shuffling are random; however, it is trivial to change it to specify where you want each type of encryption to occur. As mentioned earlier, you can also specify which types of elements are included in your decoy constructs as well as the structure and size of the constructs. You can tailor the algorithm to fit whatever problem and needs you encounter.

Another approach to explore in the future is to also place recombinase sites in ORFs themselves (separate a gene into two or more modules). The recombinase attB and attP sites are roughly 22 amino acids long so even including it in the middle of genes to separate one gene into two different modules might still work. This would give a more `scrambled' sequence that may be harder to decrypt. As recombinase sites are small, in many cases they may not affect the function of the protein that much, and therefore the network will function with recombinase sites within ORFs. A few adjustments we need to make if we implement this method are to add spacers to the recombinase sites so that it will not shift the remaining ORF out of frame and make sure the attB and attP sites do not contain any stop codons in the reading frame. In the example with using DNA as storage, if the only thing that you wanted to recover was a sequence, then you can encrypt however you want and then sequence the product and remove the known scars left by the recombinase sites to recover the original sequence. These are just a few of the possible applications for this method.

The next step for this project at this point would be to actually implement the encryption of the Voigt lab circuit construct and see if the decryption works experimentally. From the experimental data, we can further optimize the encryption algorithm to include better heuristics developed from actual result (e.g. which pairs work well together, how big of spacers are needed, etc.)

References:

1. Bradley, P. (2012) Structural modeling of TAL effector –DNA interactions. Protein Science. 21, 471-474
2. Miller, J. *et al.* (2011) A TALE nuclease architecture for efficient genome editing. Nature Biotechnology. 29, 149-153
3. Reyon, D. *et al.* (2012) FLASH assembly of TALENs for high-throughput genome editing. Nature Biotechnology. 30, 460-465
4. Stormo, G; Zhao, Y. (2010) Determining the specificity of protein-DNA interactions. Nature. 11, 751-760
5. Zhang, F. *et al.* (2011) Efficient construction of sequence-specific TAL effectors for modulating mammalian transcription. Nature Biotechnology. 29, 143-149
6. Szymczak-Workman, D. *et al.* (2012) Design and Construction of 2A Peptide-Linked Multicistronic Vectors. Cold Spring Harbor Protocols.
7. Lefrancois, P. *et al.* (2009) Effiicient yeast ChIP-Seq musing multiplex short-read DNA sequencing. BMC Genomics 10: 37.
8. Garg, A. *et al.* (2012) Engineering synthetic TAL effectors with orthogonal target sites. Nucleic Acids Research 1-12.
9. Spitz, F; Furlong, E. E. M. (2012) Transcription factors: from enhancer binding to developmental control. Nature Review Volume 13.
10. Alon, U. (2007) an Introduction to Systems Biology: Design Principles of Biological Circuits. Chapman & Hall/ CRC Mathematical and Computational Biology Series

11. Dong, D. *et al.* (2012) Structural Basis for Sequence-Specific Recognition of DNA by TAL Effectors. Science 10 February 2012.

12. Musladin, S; Barbaric, S. (2010) yeast PHO Genes: An excellent Model for Elucidation of Chromatin-Remodeling Mechanisms. Food Technology Biotechnology. 48 (3) 308-310.

13. Gerdes, S.Y. *et al.* (2003) Experimental Determination and System Level Analysis of Essential Genes in Escherichia coli MG1655. J. Bacteriol 185(19):5673.

14. Keseler *et al.*, Nuc Acids Res, 39:D583-90 2011.

15. Link, A. (1997) Methods for generating Precise Deletions and Insertions in the Genome of Wild-Type Escherichia coli: Application to Open Reading Frame Characterization. J. Bacteriol. Vol 179 p 6228-6237

16. Bonnet, J. *et al.* (2013) Amplifying Genetic Logic Gates. Science 340, 599

17. Church, G. *et al* (2012) Next-Generation Digital Information Storage in DNA. Science 337, 1628

18. Goldman, N. *et al.* (2013) Towards practical, high-capacity, low-maintenance information storage in synthesized DNA. Nature Volume 000, 1

19. Nern, A. *et al.* (2011) Multiple new site-specfic recombinases for use in manipulating animal genomes. PNAS vol 108 no. 34

20. Moon, T.S. *et al.* (2012) Genetic programs constructed from layered logic gates in single cells. Nature Vol. 491, 249

21. Benenson, Y. (2013) Recombinatorial Logic. Science Vol. 340, 554

22. Siuti, P. *et al.* (2013) Synthetic circuits integrating logic and memory in living cells. Nature Biotechnology Vol. 31 No. 5.

23. Khalil, A.S, *et al.* (2012) A synthetic Biology Framework for Programming Eukaryotic Transcription Functions. Cell 150, 647-658.

24. Ellis, T. *et al.* (2009) Diversity-based, model-guided construction of synthetic gene networks with predicted functions. Nature Biotechnology Vol. 27 No. 5

25. Ghasemi, O. *et al.* (2011) Bayesian parameter estimation for nonlinear modeling of biological pathways. BMC Systems Biology 5

26. Mukherji, S, and van Oudenaarden, A. (2009) Synthetic biology: understanding biological design from synthetic circuits. Nature Reviews Vol. 10

27. Cheng, A.A, and Lu, T.K. (2012) Synthetic Biology: An emergine Engineering Discipline. Annu. Rev. Biomed. Eng. 14:155-78

28. Sipser, M. (2012) Introduction to the Theory of Computation 3rd ed. Cenage Learning.

Appendix

Biological Circuit Engineering Teaching Experience

Another one of my responsibilities during my M.Eng studies was helping to develop the curriculum then becoming a teaching assistant for 6.S193 (BioCel), a new laboratory course about designing and building biological circuits in yeast. Professors Timothy Lu, Ron Weiss, and Rahul Sarpeshkar were the faculty in charge of the class.

The responsibilities in the fall semester involved developing the feed forward loop [24] and zinc finger synthetic transcription factor [23] modules. This required designing all the experiments described in the papers so that it was scalable to an entire class, as well as proofing protocols that could be followed by students with a wide range of backgrounds. The original strains had to be manipulated before use by the class so that it followed the flow of the course and was easier for the students to use.

When class started in the spring we decided to incorporate two computational portions, one for each module, which I designed. In the feed forward loop module, this required backwards engineering the steady state hybrid promoter model to an easier start point for the students. A write up was made to guide the students through the process of creating the final steady state hybrid model.

In the zinc finger module, we came up with the idea of having the students create their own interesting circuit simulations based on data from real parts that were characterized throughout the lab. This required a method figuring out a way to simulate biological circuit dynamics which

we did through the MATLAB SimBio library. I also provided the students with a suite of functions that could be used to extract parameters given their experimental data as input. The two programming assignments I designed start on the next page.

Code

Due to the number of different programs mentioned in this thesis. I chose to not exclude any of the actual code in this document. All requests for the source code in any of the projects should be directed at the author.

DAY 5: modeling the behavior of the Feed Forward Loop.

The feed forward loop (FFL) consists of a transcription factor X regulating a second transcription factor Y, and both X and Y regulating gene Z. You can think of this as a graph with three nodes X, Y, and Z, and three directed edges, one from X to Y, one from Y to Z, and one from X to Z, as shown in figure 1. We will not consider the cases with self-loops or backwards edges (that would make feedback loops). Since at each step in the regulation the transcription factor can either repress or activate, we have a total of $2^3 = 8$ possible types of FFLs consisting of three elements. It is a little more complicated to do the math for how many possible types of FFLs there are with more than three elements because it is not clear which element would regulate what, or how many total edges there would be.

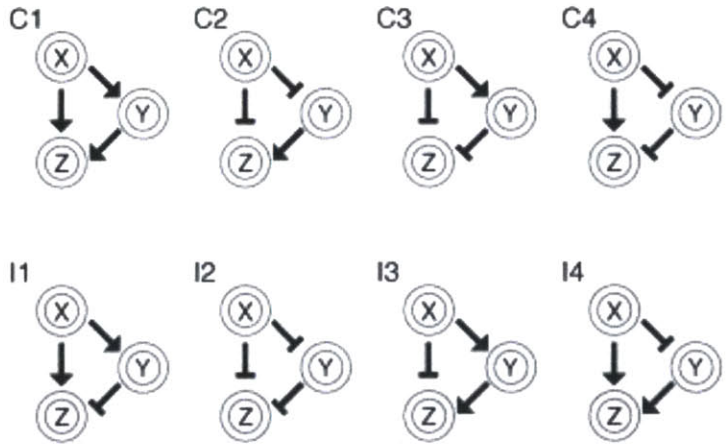


Figure 1: The eight types of feed forward loops containing 3 elements. The coherent loops are in the top row and the incoherent loops are in the bottom row. A pointed arrow indicates activation while a flat one indicates inhibition.

The eight types of three element FFLs can be classified into two groups: coherent and incoherent. This grouping is based on comparing the function of the direct path from X to Z to the function of the indirect path that goes through Y. For example, in C1 in figure 1, the X is an

activator for Z directly and X is also an indirect activator for Z since X activates Y and Y activates Z. Contrast this with I1, X is an activator for Z directly but X is a repressor for Z indirectly since X activates Y but Y represses Z. In coherent FFLs (CFFLs), the direct and indirect path from X to Z have the same function while in incoherent FFLs (IFFLs); the direct and indirect paths have opposite functions. The circuit you are building has TetR as X, LacI as Y, and GFP as gene Z.

Question 0.1: Which FFL (C1-C4, I1-I4) corresponds to the system you are building?

Make sure you answer all questions in your lab notebook for your write-up of today. You may be quizzed on the material in this exercise during your oral presentation for module 1.

Recall the structure of the IFFL we have been building in class (figure 2):

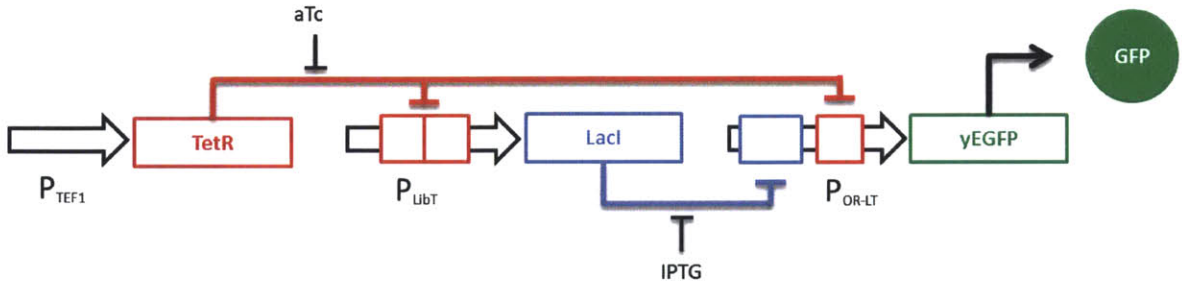


Figure 2: The block diagram of the entire IFFL. LacI is TetR-regulated and a hybrid promoter containing both TetR and LacI binding sites controls the expression of yEGFP.

The computational modeling of this interaction is rather complex since both LacI and TetR repress the hybrid promoter (P_{OR-LT}) controlling GFP expression levels. The total concentration of LacI is dependent on the total concentration of TetR since the promoter of LacI (P_{libT}) has two TetR binding sites as shown in figure 2. Those two characteristics make this circuit an IFFL.

Note that even though gene expression stochasticity in the synthetic gene networks results in some interesting experimental observations, we mainly focus on average population behaviors, both temporal and steady state.

In this lab we will start by modeling simpler networks and build up to modeling the entire IFFL. In part 1, you will be given MATLAB code that models two independent systems. The first system has TetR, controlled by a TEF1 promoter (constitutive promoter), repressing the expression of GFP. This system is inducible with anhydrotetracyclin (aTc). The second system has LacI, also controlled by a separate TEF1 promoter, repressing expression of RFP. This system is inducible with IPTG. Refer to Figure 3 for a diagram. This is the simplest case and avoids the intricacies of both the hybrid promoter controlling output and the feed forward loop.

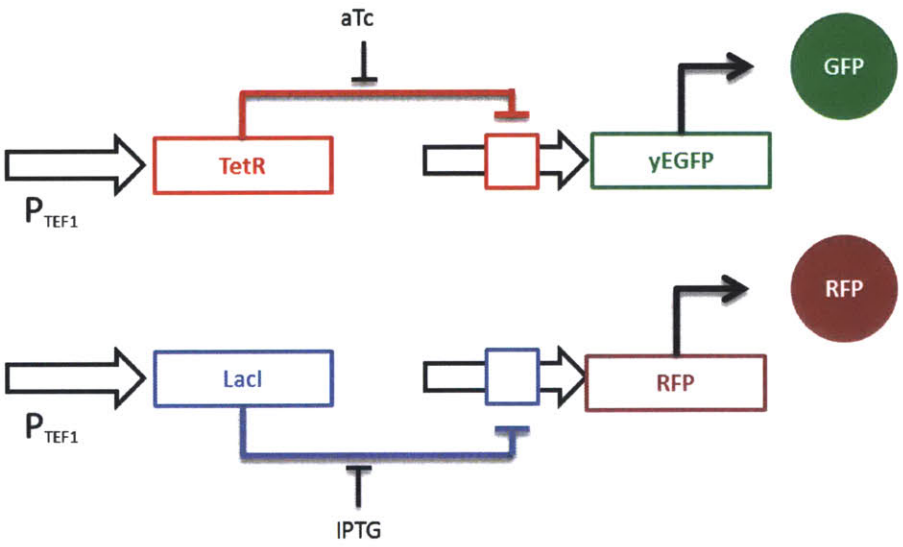


Figure 3: The block diagram of the two independent systems in part 1. TetR represses GFP and LacI represses RFP.

In part 2, you will build on the model from part 1. TetR and LacI are still both controlled by constitutive TEF1 promoters but the output is GFP controlled by a hybrid promoter that is

repressed by both TetR and LacI, as shown in Figure 4. By the end of part 2, you will have modeled dynamics of $P_{OR\text{-}LT}$.

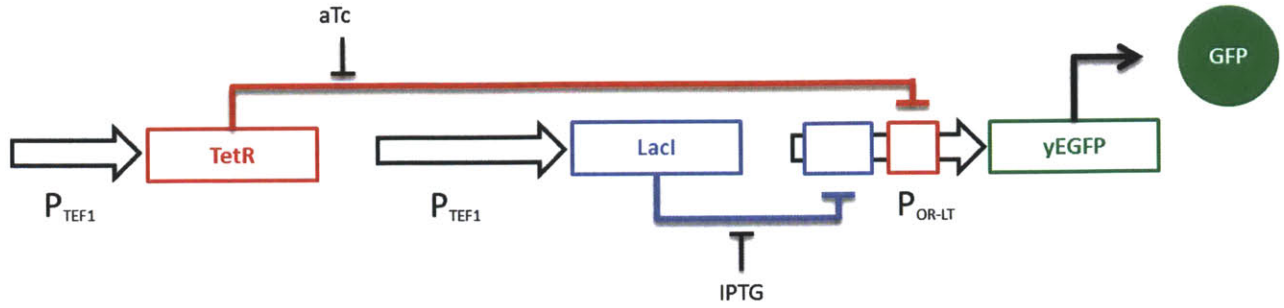


Figure 4: The block diagram for the system in part 2. Constitutive TEF1 promoters control both LacI and TetR. A hybrid promoter containing both TetR and LacI binding sites controls expression of yEGFP.

Finally, part 3 builds on the model from part 2. The promoter controlling LacI expression will no longer be the constitutive TEF1. Instead, it will be one of the promoters from the promoter library you worked with in lab (the TX–T20 promoters). Refer back to Figure 2 for a diagram. All these promoters are repressed by TetR and inducible with aTc. However, maximum expression and basal expression levels differ for each promoter. On day 2 of the class you started cultures of each promoter in various media in order to measure those parameters. On day 3, the TAs measured the GFP levels using flow cytometry for you, providing a read out of basal and maximally induced expression levels from each promoter. You need to use this data to determine every promoter's minimum (TetR repressed) output (S_{min}) and maximum (TetR unrepressed) output (S_{max}). These values are inputs for your program, which you will use to calculate efficiency of the inhibited and uninhibited gene, LacI total concentration, and expected GFP output levels. Keep in mind that the ability of TetR to repress the P_{LibT} maybe be quite different than its ability to repress $P_{OR\text{-}LT}$. The purpose of this exercise is to create a model from pieces so we assume that the transcriptional rates can transfer but this may not be the case.

Part 1

Please open MATLAB on your computer and open the file IFFLss.m in the file editor. The only thing you need to do for part 1 is execute the file within Matlab. You can do this by changing the directory to where the file is saved and typing IFFLss into the command window and hit enter to run. Alternatively, you can run the file by either pressing F5 or the Run button on the top toolbar in the editor window. The output should look like figure 5.

Question 1.1: Include snapshots of your own execution of the program in your lab write-up. Please include documented code, snapshots of your output, and analysis of your results for all future parts.

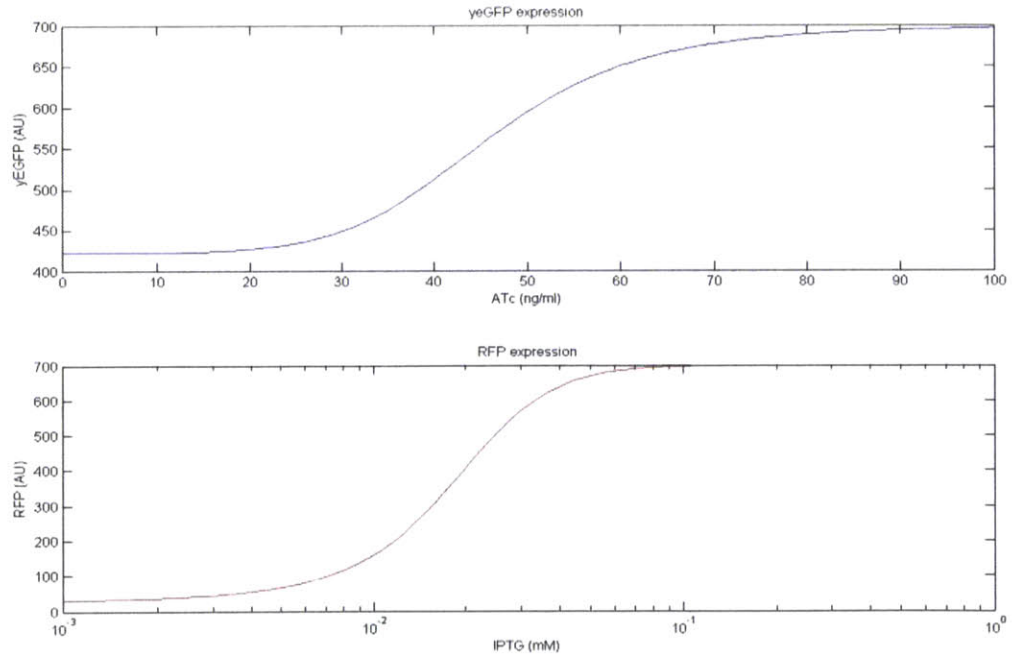


Figure 5: The output of executing IFFLss.m. The top graph shows yEGFP output (AU) vs. aTc concentration (ng/mL). The bottom graph is expected yEGFP output (AU) vs. IPTG concentration (mM).

Now let's delve into the code to see what is actually going on. The constants section assigns numerical values to variables that you will need in your model. These numbers were determined experimentally. We will go more into what these variables represent in a little bit. The transcriptional rates (TX*i*'s) refer to different states that the two individual systems can be in, as shown in figure 6. State S1 corresponds to no LacI binding to the LacI regulated promoter, state S2 corresponds to a LacI binding. State S3 corresponds to no TetR binding to the TetR regulated promoter and state S4 corresponds to TetR binding. One thing to note is that since there are two independent systems, states S1 and S2 do not affect states S3 and S4.

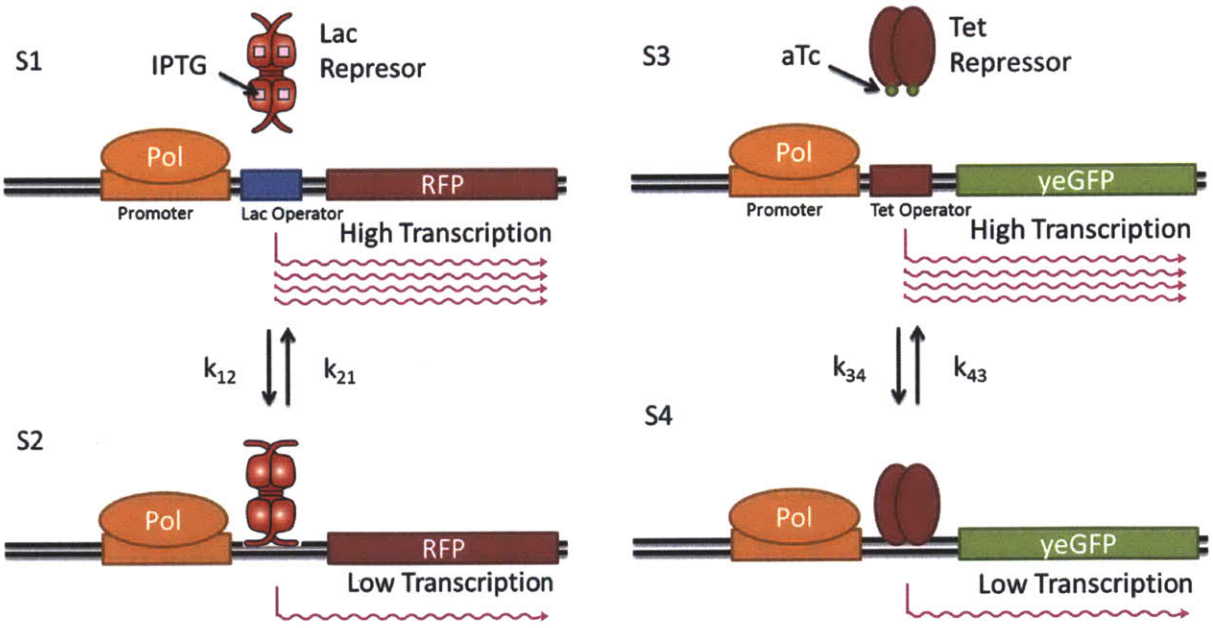


Figure 6: Diagram of the possible states of the promoters expressing GFP / RFP. The two systems are independent.

In the equations section of the code, the first few lines initialize vectors for the different inputs and outputs. Since the LacI-regulated and TetR-regulated promoters are characterized similarly, we'll explain the technical details in parallel. The range of the concentration of aTc used is 0 -

100 ng/mL. If you look back at figure 5, you will see that from our model, the GFP curve begins to saturate around 100 ng/mL aTc but is not completely saturated.

Question 1.2: What does it mean if the curve is not completely saturated? What concentration did you use in your actual experiment? Play around with the code and plug in different values to see how that affects the output.

The range of IPTG concentrations is slightly different. It is measured in mM and ranges from 0.001 - 1 mM. These ranges were optimized to capture the interesting part of the induction.

To actually calculate the expected expression level of the fluorescent protein output, we need to quantify the relationship between inducer concentration and the active repressor concentration. A general Hill function can be used to quantify the fraction of inactive repressors. For example, $f_{IPTG} = [IPTG]^{ni} / (K_{IPTG}{}^{ni} + [IPTG]^{ni})$ represents the fraction of inactive LacI monomers (those bound by IPTG). K_{IPTG} is related to the dissociation constant between LacI and IPTG, and it represents the cooperativity between IPTG and LacI. The exponent ni is the Hill coefficient for IPTG and was experimentally determined. The equation for f_{aTc} follows the same format.

Question 1.3: Both expressions for the inactive repressor fraction are assumed to be independent of repressor (TetR/LacI) abundance. Why can we do this?

Next, we have four constants: k_{12}, k_{21}, k_{34}, k_{43}, which represent the rate constants for the transition between two states (refer back to figure 6). In addition, k_{12} is the association rate ($k_{AssociationL}$) between tetramers of LacI and DNA, k_{21} is the dissociation rate ($k_{DissociationL}$)

between tetramers of LacI and DNA, k_{34} is the association rate ($k_{AssociationT}$) between TetR dimers and DNA, and k_{43} is the dissocation rate ($k_{DissociationT}$) between TetR dimers and DNA. We define $k_{EL} = k_{AssociationL} / k_{DissociationL}$ and $k_{ET} = k_{AssociationT} / k_{DissociationT}$. We will come back to this definition of k_{EL} and k_{ET} later but in the meantime we can calculate k_{EL} another way.

The definitions of k_{EL} and k_{ET} are great if you just consider LacI tetramer and TetR dimer interactions with the promoter, but because we need to relate the equilibrium constants of the promoters to monomer concentrations of the repressors, things are a bit more complicated. There are several equilibria between a repressor monomer and an activated oligomer. First the repressor has to be unbound from its inducer molecule then the repressor has to either tetramerize or dimerize, we will use one variable to account for all stages. Let k_{al} represent the lumped parameter for LacI-DNA association equilibrium and k_{at} represent the lumped parameter for TetR-DNA association equilibrium. The constants k_{el} and k_{et} can be calculated by multiplying the association equilibrium by the abundance of active LacI and TetR proteins (those not bound by IPTG or aTc) respectively. You can write this in equation form as: $k_{EL} = k_{al} ((1 - f_{IPTG})[LacI])^4$ and $k_{ET} = k_{at} ((1- f_{aTc})[TetR])^2$. Since total TetR and LacI abundance is assumed to be constant in this specific case, we can combine it with the association equilibrium to create parameters k'_{al} and k'_{at}. As you will see in part 3, this is not true if the total abundance were not constant. The equations become $k_{EL} = k'_{al} (1 - f_{IPTG})^4$ and $k_{ET} = k'_{at} (1- f_{aTc})^2$, where lumped parameters k'_{al} and k'_{at} already include the total repressor abundance.

Question 1.4: Why is the exponent of k_{EL} to the fourth while k_{ET} is only squared?

Finally, to determine how much GFP (or RFP) is produced, we need to compute the probability that the promoter is at a certain state (S1/S2 or S3/S4) at any given time. The state transition is essentially the evolution of a two-state Markov chain. A Markov chain is a mathematical model that contains different states and transition probabilities between them. It is especially useful when finding the steady state probability of being at a certain state. We will use the notation P*i* to denote the probability of being at state S*i*. For example, in the TetR system, the probability of the promoter at each of the states can be described using the following system of equations:

1. $P3 * k_{34} = P4 * k_{43}$
2. $P3 + P4 = 1$

In steady state, by definition, the rate at which promoter transitions from unbound to bound state ($P3 * k_{34}$) equals the rate at which it transitions from bound to unbound ($P4 * k_{43}$). And since the promoter must be either in the unbound state or the bound state, P3 + P4 =1. If we substitute in the fact that $k_{34} = k_{AssociationT}$ and $k_{32} = k_{DissociationT}$, we can write equation 1 in terms of k_{ET} (recall $k_{ET} = k_{AssociationT}/ k_{DissociationT}$):

3. $k_{ET} * P3 = P4$

Solving the above, we find that $P3 = 1/ (1 + k_{ET})$ and $P4 = k_{ET} / (1 + k_{ET})$. GFP production is described by the expression: *effg* * (P3*TX3 + P4*TX4). Here *effg* is the fluorescent protein production efficiency, P*i* is the probability of being at state *i* and the TX*i* is the relative transcription rate at state *i*. Make sure you understand all the concepts and calculations from this part, as well as the simulation code, before moving on to part 2.

Part 2

Now we will build on the code from part 1. In part 2, LacI expression is still controlled by the TEF1 promoter. However, LacI is no longer an independent repressor just regulating expression of RFP; instead, together with TetR, it regulates expression of GFP. We introduce a hybrid promoter that has both LacI and TetR binding sites, acting as an NOR gate controlling expression of GFP. Note that the promoter functions as a NOR gate with respect to inputs LacI and TetR (either LacI or TetR will repress) but functions as an OR gate with respect to inputs IPTG and aTc (either IPTG or aTc will activate) within the context of the entire circuit when LacI and TetR are expressed constitutively (Figure 4).

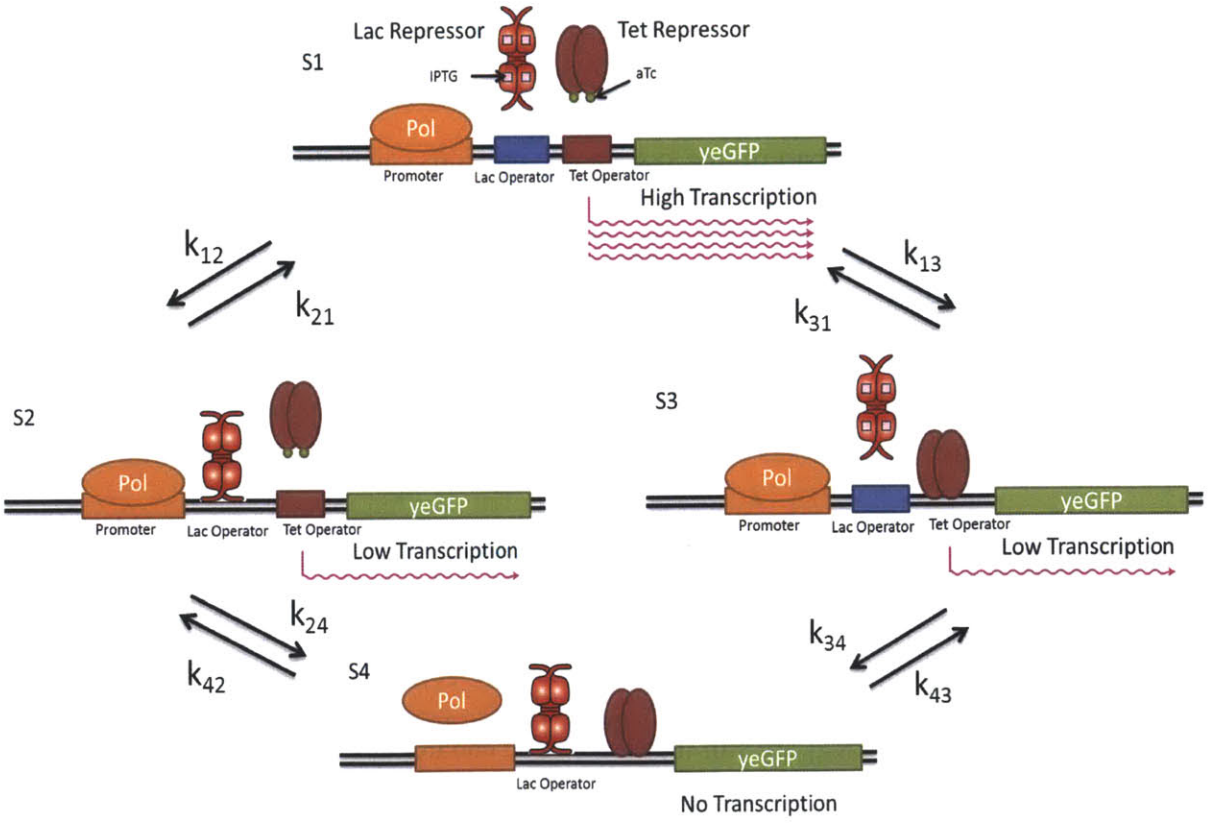


Figure 7: Diagram of four possible states of the OR-gate promoter in the negative feedforward loop network.

The first thing we need to do is redefine the states S1-S4 to reflect the behavior shown in Figure 7. State S1 still corresponds to no repressor binding to the hybrid promoter, S2 corresponds to only LacI binding, S3 corresponds to only TetR binding, and S4 corresponds to both LacI and TetR binding. The new transcriptional rates have been experimentally determined to be the following:

TX1 = 1, TX2 = 0.02, TX3 = 0.6, TX4 = 0.01.

Question 2.1: Do any of these transcription rates seem peculiar to you? If so, hypothesize why this might have occurred. In particular, comment on the value of TX3 – what does it tell you about the hybrid promoter?

Since the four states are now part of one system, the previous calculations of P1-P4 do not hold anymore. You will need to recalculate the probabilities similar to the method used above in part 1, except now it is a four-state Markov chain instead of two states.

Question 2.2: What are the new rates k_{12}, k_{21}, k_{13}, k_{31}, k_{24}, k_{42}, k_{34}, and k_{43} in terms of $k_{AssociationT}$, $k_{DissociationT}$, $k_{AssociationL}$, and $k_{DissociationL}$? Also write out the new four-state steady state Markov chain equations for the system.

The definitions of K_{EL} and K_{ET} are still the same. You can use any symbolic manipulator to find the equations for the new probabilities P1-P4 in terms of K_{EL} and K_{ET}. We will provide them to you but feel free to try to derive it yourselves.

$$P1 = \frac{1}{k_{EL}k_{ET}+k_{EL}+k_{ET}+1}$$

$$P2 = \frac{1}{k_{ET} + 1 + \frac{k_{ET}}{k_{EL}} + \frac{1}{k_{EL}}}$$

$$P3 = \frac{1}{k_{EL} + \frac{k_{EL}}{k_{ET}} + \frac{1}{k_{ET}} + 1}$$

$$P4 = \frac{1}{\frac{1}{k_{ET}} + \frac{1}{k_{EL}} + \frac{1}{k_{ET}k_{EL}} + 1}$$

Question 2.3: Verify that P1 + P2 + P3 + P4 sums to 1.

The format of your output should be a 2D matrix instead of a 1D matrix mapping GFP output levels to the combination of concentrations of aTc and IPTG as in part 1. You can use the following code snippet to plot the 3D graph of your output if you would like, or come up with a better way of displaying the data.

Code:

```
figure()
surf(yEGFP)
shading flat; colormap(jet); axis([0 100 0 100 0 700]);
set(gca,'FontSize',12,'XTick',(0:100/3:100),'XTickLabel',{'10^0','10^-
1','10^-2','10^-3'})
ylabel('aTc (ng/ml)'); xlabel('IPTG (mM)'); zlabel('yEGFP (AU)');
title('yEGFP Expression');
```

Question 2.4: Interpret the results of your model; is it what you expected?

Question 2.5: Include a commented version of your code as well as graphs you generate.

Part 3

Finally, we will complete the IFFL model by incorporating the feed forward aspect. In the FFL, the TEF1 promoter constitutively drives TetR production, and the TetR-regulated promoter with two tetO2 (TetR binding) sites controls LacI production. Since LacI is now regulated by TetR, the behavior will change based on which TetR-regulated promoter from the library is used. We need to recalculate the LacI equilibrium constant, k_{EL}, to include this change. The variable k_{al} right now has LacI abundance fixed at 15000, but you will need to change this in the new calculation of k_{EL} to account for different total concentrations. To find LacI abundance, we first need an expression for the probability of the unoccupied TetR-regulated promoter (i.e. no TetR repression):

4. $$P_{e,tet} = \frac{1}{1+\frac{(1-faTc)^4[TetR]^4}{K_{DT}^2 K1 K2}+\frac{(1-faTc)^2[TetR]^2}{K_{DT}K1}}$$

K_{DT} is the constant for TetR dimerization, K1 is the constant for TetR dimer – DNA interactions, and K2 is the constant for TetR dimer – DNA and TetR dimer – previous TetR dimer interactions. Refer to figure 8 for a visual representation. The derivation of equation 4 is beyond the scope of this course so we will not go further into it.

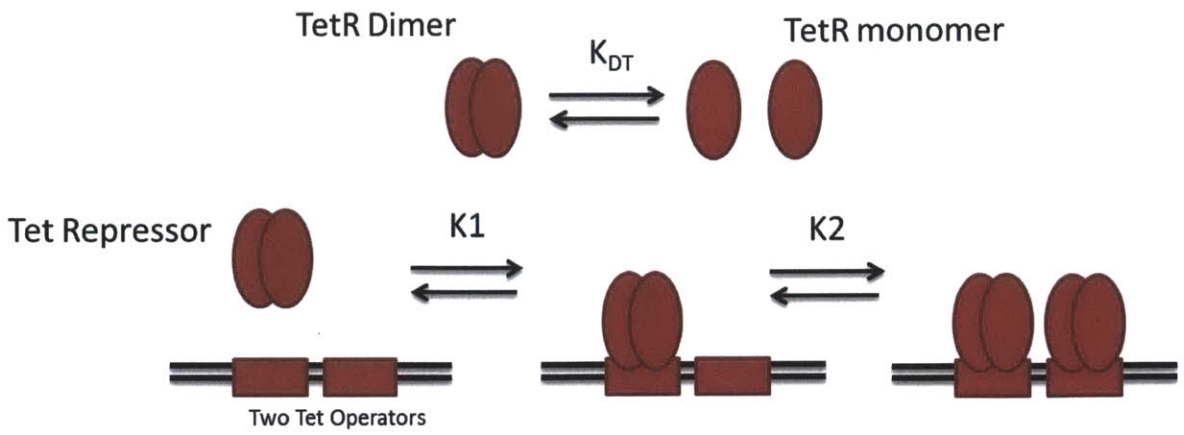


Figure 8: The different constants relating to TetR binding to promoter.

We can use lumped non-dimensional parameters *a* and *b* representing $[TetR]^2 / (K_{DT}K2)$ and $[TetR]^2 / (K_{DT}K1)$ respectively to simplify the equation. The parameters were experimentally estimated to be $a = 464$ and $b = 768$. You can convert equation 4 to use parameters *a* and *b* to calculate the probability of an unoccupied TetR-regulated promoter. The probability of having an unrepressed promoter directly affects the synthesis strength of the downstream genes (in this case, LacI). The steady state protein abundance of a downstream gene is $eff * p + eff * cr * (1-p)$, where *eff* is the transcription and translation efficiency of the gene, *p* is the probability of the operator being empty, and *cr* is the relative synthesis efficiency when the operator is occupied. The eff * *p* term corresponds to efficiency when operator is empty and the $eff * cr * (1-p)$ term corresponds to efficiency when repressor is bound to operator.

The values of *eff* and *cr* will differ based on your experimental results.

Question 3.1: How do you map the input values S_{max} and S_{min} to *eff* and *cr*.

With your calculations for the total concentration of LacI along with your new k_{EL}, you can figure out your final expected GFP output similar to the method used in part 2. Your final MATLAB program should take inputs S_{min} and S_{max} and the actual shape of your 3D plot will change based on S_{min} and S_{max}. One preprocessing step you need to do is normalize your experimental values of S_{min} and S_{max}. The output of FACS is in arbitrary units (A.U.). The absolute value for each S parameter provided by the machine may vary across different FACS machines or different users but their relative values to a common control sample should be conserved and independent of user and machine. In order to get around this, normalize your data so that the TX promoter has a S_{max} value of 1000. Plug in the values from your characterization data of T1-T20 and TX and see if it follows the trend of your experimental values.

Question 3.2: Did it follow the trend? If not, then explain why you think this is happening.

Question 3.3: Analyze the overall performance of your model, what can you do to improve it.

Question 3.4: Include a commented version of your code as well as graphs you generate.

6.S193: Open Ended Design Project

April 26, 2013

Overview

An important goal for synthetic biology is designing interesting circuits from characterized parts. For the computational part of module two, you are assigned an open-ended design project that explores this concept. You will have to gather experimental results from this module and the last one; determine the kinetic properties of the transcription factors that we analyzed; and design, simulate, and analyze a new circuit of your choice using only these transcription factors and relevant inducer molecules.

You will have components and data for everything you have encountered this semester. You have all the data from the IFFL characterizations already (two inputs: aTc and IPTG and two repressors: TetR and LacI). In addition you will characterize the zinc finger constructs we are working with in this module, which are activators. An important thing to consider with these zinc fingers is their orthogonality to each other and how specific each pairing is. You will generate an orthogonality matrix experimentally to account for this as shown below.

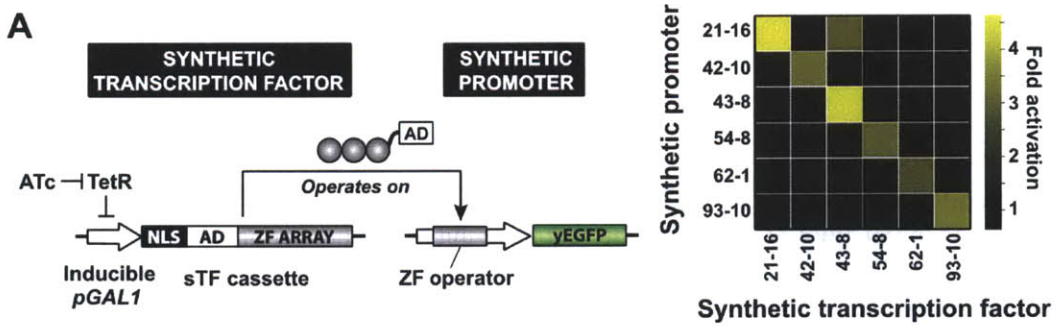


Figure 1: Zinc finger transcription factors for module II.

For promoters, in addition to the IFFL library of promoters, you can use several constitutive promoters with rate constants of your choosing.

From your input/output experimental data (as well as additional data we will provide), you can perform parameter estimation on the following

- aTc ⊣ TetR ⊣ GFP
- IPTG ⊣ LacI ⊣ GFP
- aTc ⊣ TetR ⊣ LacI ⊣ GFP
- IFFL
- aTc ⊣ TetR ⊣ ZF → GFP

Finally, in addition to the parts mentioned above, you can use two 'virtual' repressors R1 and R2. For R1, repression kinetics are similar to TetR, but there is no interaction with aTc. R2 repression kinetics are similar to LacI, but similarly there is no interaction with IPTG.

The rest is up to you. You will need to incorporate your Open Design Project into the Module II presentation, which will take place during the last class/lab (May 15). You will need to submit your (well-documented) code to us as part of the presentation, but will not be required to provide a separate writeup.

The following sections go over an example of simulation of circuit dynamics and parameter estimation.

Simulation of circuit dynamics

In the first Open Ended Design Project session we went over how to simulate a network in MATLAB using the sim biology toolkit. In particular, we covered a cascade with equation and output as shown below in Figure 2. Refer to the MATLAB code cascade_sim.m for more details.

$$pC \rightarrow R1 \dashv R2 \dashv R3$$

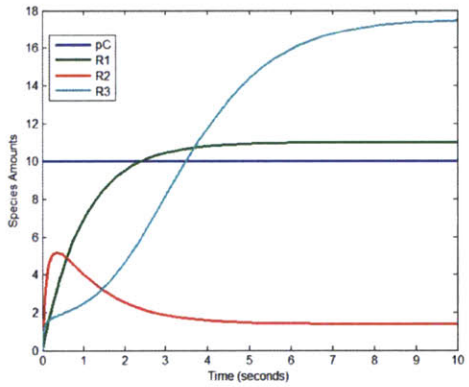


Figure 2: The equation for the cascade as well as a time series simulation of all reactants involved. The reactant pC has an initial amount of 10 and the rest have initial amounts equal to zero.

In this handout we will look at a much simpler system but also introduce parameter estimation so you can start building up your own library of characterized components. The system we will look at is the inducible interaction between aTc and GFP: aTc -> GFP. Refer to the MATLAB code induction_sim.m for more details. We will use the same blackbox approach we took last time to model this system. The two main "reactions" are:

1. aTc → aTc + GFP
2. GFP → null

The first equation will be governed by the 'Hill-Kinetics' law (type sbiowhos for a complete list of all defined abstract kinetic laws, built in and user-defined), with parameters Vm, Kp, and n. The second equation will be governed by the 'MassAction' law, with parameter k_GFP_deg. It is important to note that the order that you set these parameters when associating to kinetic law is important but somewhat arbitrary so make sure you check the reactions with the code snippet below to make sure your reaction equations are correct.

```
disp('Model reactions:');
get(modelObj.Reactions, {'Reaction', 'ReactionRate'})
```

An example simulation of this system can be seen in Figure 3 below. Make sure you understand what all the lines of code mean so that you can build your own network simulations for your project. To make a transfer function, you can change the levels of aTc then map the GFP level at steady state to those aTc values to get a GFP vs aTc transfer function graph.

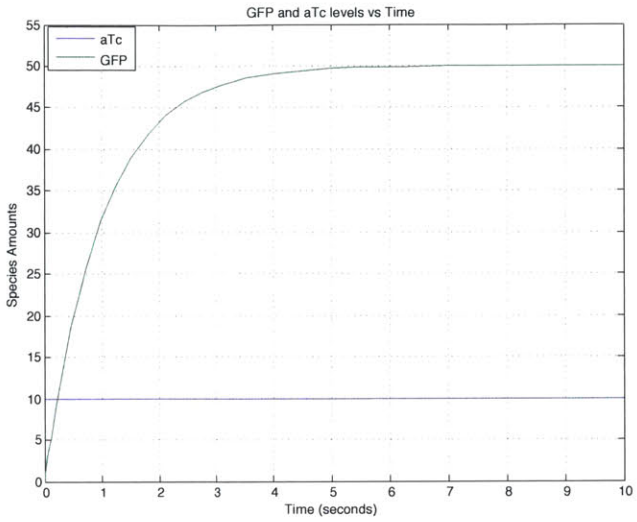


Figure 3: Simulated GFP output as a function of time. Parameters: Vm = 50, Kp = 0.02, n =2, k_GFP_deg = 1. The level of aTc is constant at 10.

Parameter Estimation

By now you should know how to generate data if you have parameters. But what you really want is to extract the parameters given experimental data. The experimental data is easy to measure but measuring the parameters is difficult and usually done indirectly through parameter estimation techniques. For our example, we will take the data from Figure 4 to estimate the parameters Vm, Kp, n, and k_GFP_deg for the actual system.

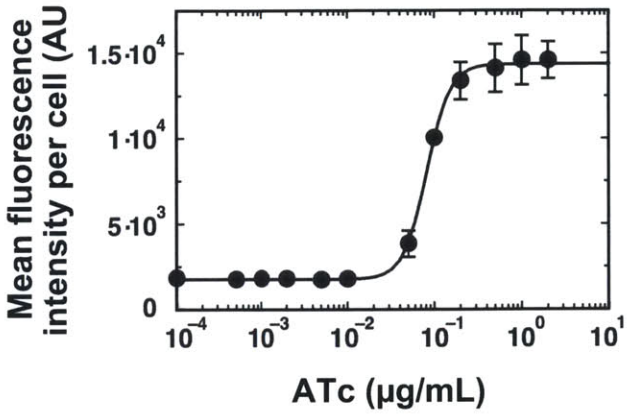


Figure 4. GFP vs aTc transfer function from the ZF paper. We will use this data to determine the parameters of the aTc → GFP reactions.

This handout will cover two different techniques of parameter estimation. They should technically do the same thing but do have different pros and cons. One is an analytical method, which requires you to fit the data to a known equation which you must be able to derive and the other is simulation based, which does not require you to know the equation but is more computationally intensive and perhaps less accurate.

Analytical

The analytical method is very straightforward and easy to implement. The caveat is that you need to know the reaction equation beforehand as well as initial guess for the parameters. This method is basically a nonlinear fit of the experimental data with an equation that incorporates the parameters you are trying to estimate. It uses a form of gradient descent so it is very sensitive to initial guesses for the parameters (i.e. bad initial guesses will yield incoherent results). This approach is great if you know how your reaction will behave and a general ballpark estimate of what your parameter values will be. Another thing to watch out for is that the nonlinear fit seems to have trouble when there is basal level expression. In this example, we will avoid such problems by using the equation form:

```
(Vm*S^n / (s^n) + Kp^n)+ basal level)
```

Refer to the code hillfit.m for more detail. The output of the parameter estimation from this method can be seen below in Figure 5.

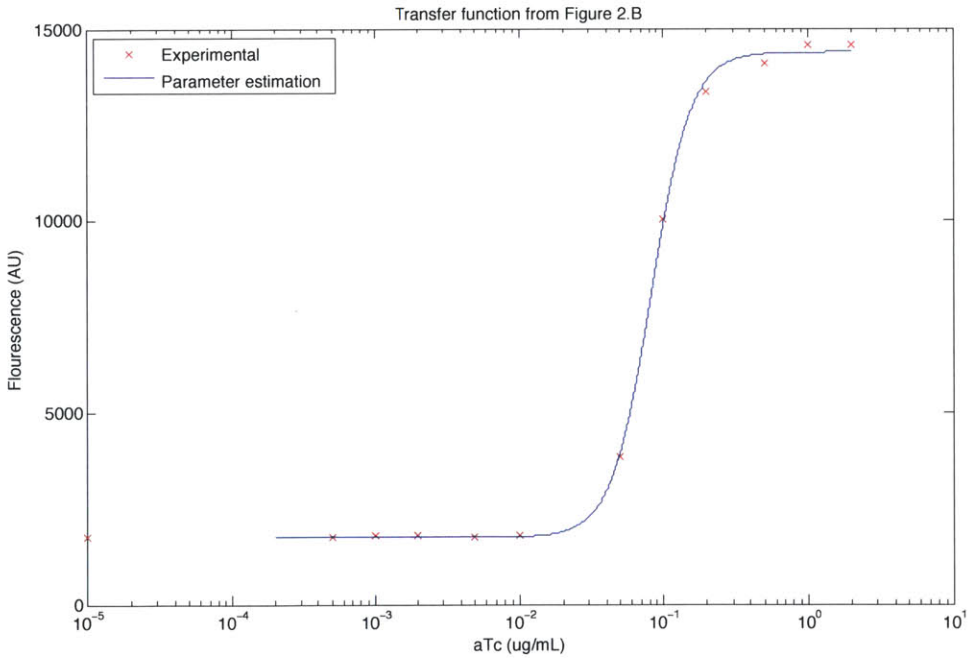


Figure 5: The hill function curve generated from the estimated parameters along with the actual experimental data points.

Simulation based

The second method relies solely on computer simulations. This approach requires no analytical derivation of equations that describe the system, which is sometimes overly cumbersome or impossible.

All we need is the sbioparamestim function in the Sim Bio toolbox. The tradeoff is more computation time and perhaps less accuracy.

The key to this approach is manipulating sbioparamestim to match the form of our experimental data. The function sbioparamestim takes in an sbio model as input. It is meant to match observed time-series data with the simulation and output parameters of a model that minimizes error. However, our observed data is not in the time domain but is steady state aTc vs GFP. We can get around this by increasing the level of aTc in the time domain in steps corresponding to the aTc levels we have experimental data for, and waiting for GFP output to reach steady state. This manipulation can be seen in Figure 6. Refer to the code paramestim.m for more details and look at the documentation for the function sbioparamestim.

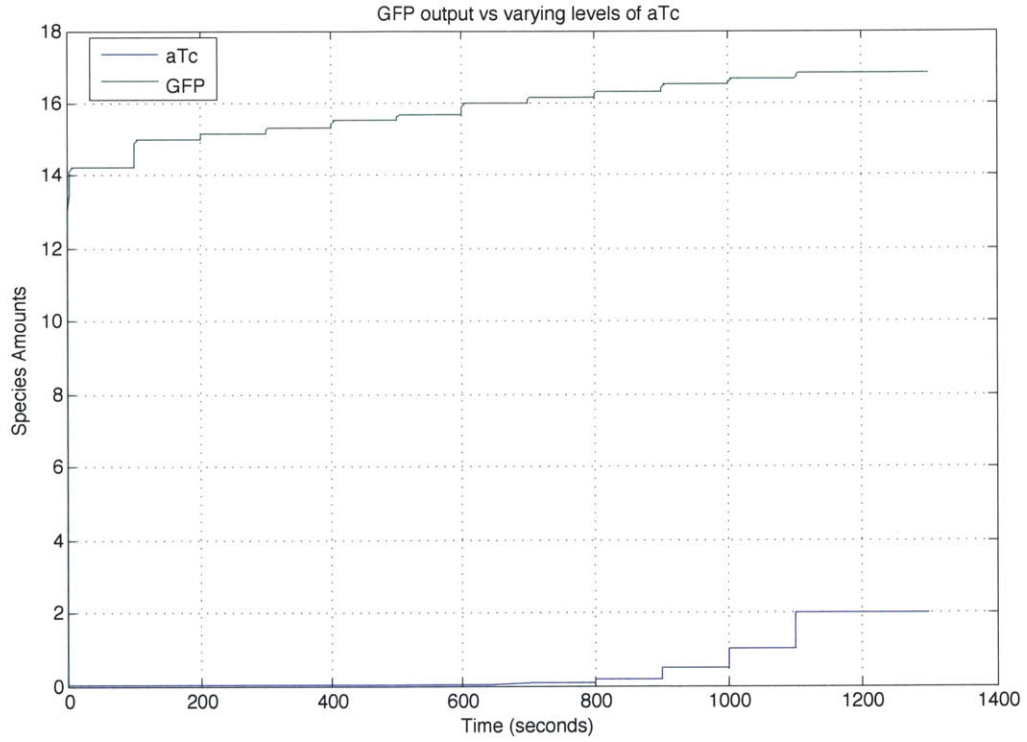


Figure 6: aTc levels vary with time and the wait is long enough for GFP to reach steady state. The GFP levels at the time corresponding to just before the next step in aTc is the steady state GFP expression level for a certain level of aTc.

One important feature here is the algorithm you select when you run sbioparamestim. Based on what your experimental data looks like, the output of each algorithm can be wildly different as shown in Table 1 for the estimation of the different parameters in this example.

Algorithm	fminsearch	lsqcurvefit	lsqnonlin	fmincon	patternsearch	patternsearch_hybrid	ga	ga_hybrid
deg	-0.0013	0.002459	0.002459	0.000988	1.1875	0.000780538	0.025696	0.007793
Vm	17.4654	58.76645	58.76645	17.62828	51954	266.8236485	2.131295	1.876367
Kp	0.0528	9.83E-04	9.83E-04	1.682013	0.348125	0.129433989	0.21097	0.164892
n	1.4833	1.328133	1.328133	0.000207	2	8.810381816	0.018238	0.333855

Table 1: The output of the parameter estimations using different algorithms, the default algorithm is lsqnonlin (Which also happened to perform best in this case).

And finally you can see the output curves generated from this technique in Figure 7 below.

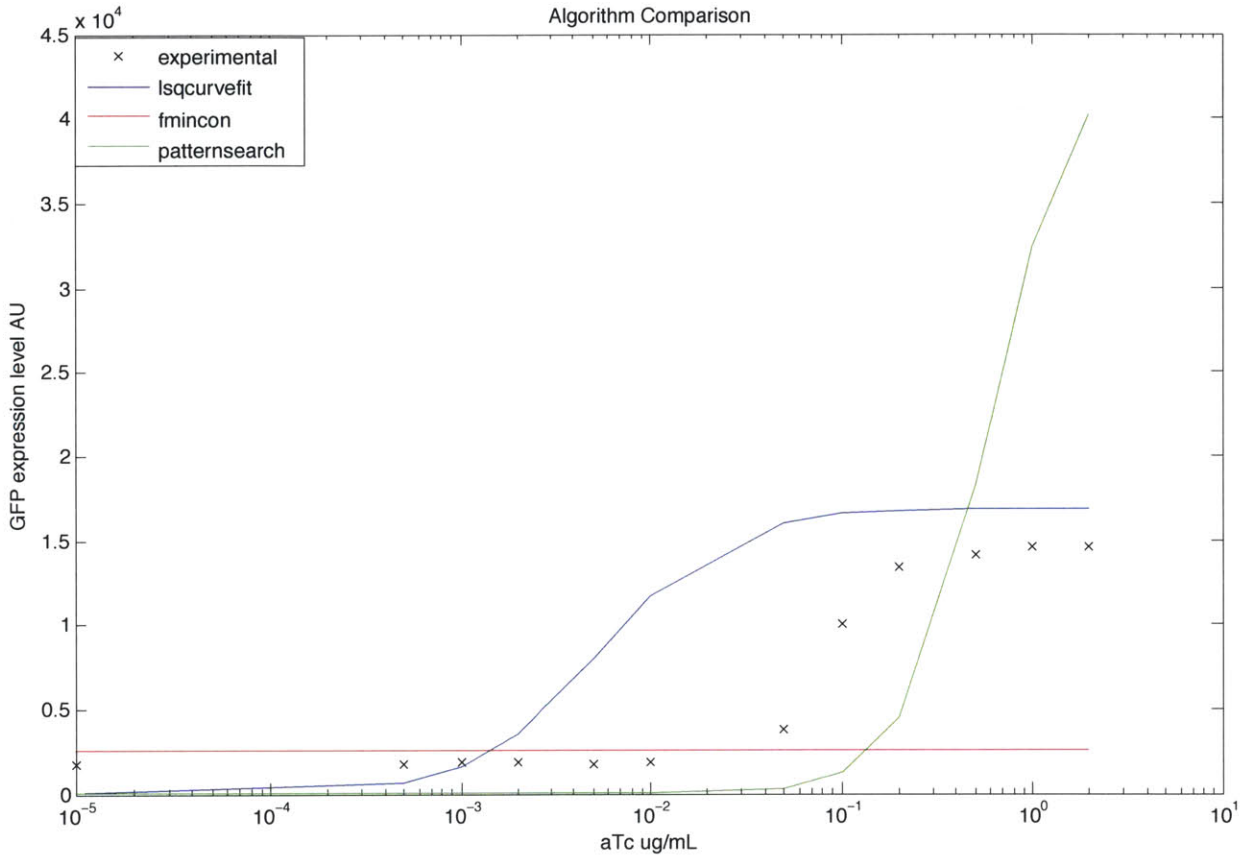


Figure 7: The hill function curve generated from the estimated parameters with different algorithms.

This technique is more flexible than the analytical one but is less accurate in this instance. It seems to be very sensitive to spacing of the data points in the x-axis. Both approaches have their pros and cons. You can choose one or the other, combine and improve the two approaches, or even come up with your own parameter estimation functions for your Open Ended Design Project. One method we would recommend that combines the strength of both approaches is running the simulation method to get rough values for the parameters, and then use those values as the initial guesses for the nonlinear function fit in the analytical method. If you are so inclined, you may want to examine why sbioparamestim is less accurate than the analytical method (for example, how does sbioparamestim currently compute its error function to quantify deviation from desired data points, and if the method is suboptimal, can you fix that?).