AN INVESTIGATION OF IN-SILICO EVOLVING NETWORKS

By

VERA ELIZABETH THORNTON

_____

A Thesis Submitted to The Honors College

In Partial Fulfillment of the Bachelors degree
With Honors in

Mathematics

THE UNIVERSITY OF ARIZONA

MAY 2017

Approved by:

_____

Dr. Kevin Lin
Department of Mathematics

Abstract

This honors thesis project builds on the work of Paul Francois, Eric Siggia, and Vincent Hakim that uses in-silico genetic network models to study evolution. Genetic networks are represented as systems of ordinary differential equations (ODEs). Each term in the differential equation corresponds to a chemical reaction, and the solution of the ODE, found using the fourth order Runge-Kutta algorithm, yields the levels of proteins in the system over time which is due to the expression of genes in the network. After replicating specific networks, a more general program was written which began with a very simple network and evolved to a more complex system through successive rounds of mutation and selection. Selection was achieved by solving for the levels of protein over time, and then scoring the network based on how closely these levels approached the desired behavior profile. A genetic network model similar to one presented in the original papers was successfully evolved. Then the same program was used to attempt to evolve an oscillating network system.

Acknowledgements

Many thanks to Dr. Kevin Lin for his mentorship and guidance which were critical to the success of this project. In addition, we would like to thank Dr. Francois for generously making a version of the evolving network code available for comparison with the code created in this project. Finally, we would like to recognize all the authors on the series of papers on evolving networks for their contributions to this new area of biological modelling.

1 Introduction:

A challenge in the field of evolutionary biology is determining how the structures we observe in nature evolved to their current form. One especially challenging aspect is deciphering the relationship between the genotype, an organism's genetic code, and the phenotype, the organism's physical characteristics. It is well known that the genotype codes for proteins, but this is only the beginning. Many basic biological functions depend on finely tuned fluctuations in protein levels. This can be achieved by proteins that form complexes with other proteins, proteins that degrade other proteins, or allowing one protein to promote or repress the genes that code for another protein. These interactions that modulate the relationship between the genetic code and its expression in the phenotype is complex and still imperfectly understood.

Recently, the use of computer based models has made possible an experimental approach to studying evolution. Instead of studying a system as it functions in a living cell (in vivo), or attempting to recreate it using proteins in a test tube (in-vitro), the system can be modelled using a computer program (in-silico). Unlike in-vivo and in-vitro methods, a computer program offers the possibility to simulate tens of thousands of generations of mutation and selection in minutes or hours, as opposed to millenia. Another challenge of working from a living organism back to an abstracted model is complexity. A living system is incredibly complex, and it is not always clear what aspects should be incorporated in a model and which can be abstracted away for the sake of simplicity and clarity. The evolution in-silico approach applies fundamental principles of evolution by which the phenotype (here simulated levels of protein) is selected upon leading to cumulative effects on the underlying genotype. By simulating

multiple generations of mutation and selection, a model genetic network with the key genes and interactions needed to express the desired behavior can be arrived at. In turn, this model network can be studied to gain new understanding of the complex living system.

The researchers Dr. Paul Francois, Dr. Eric Siggia, and Dr. Vincent Hakim, along with various collaborators, have published a series of papers describing the methods by which they achieved computer based models of networks capable of basic yet vital behavior through a process of successive mutation and selection. These models include such critical functionalities as a bistable switch, a network with levels of protein that spike and then adapt in response to an input protein, and a network that develops precise oscillations.

The goals of this honors thesis are:

1. To replicate some of the final networks described by Francois, et al in order to understand the implementation of the model
2. Create a basic mutation and selection program and evolve a new network through successive mutations
3. Determine if, given the right fitness criteria, an oscillating network can be arrived at through successive mutations using only basic protein interactions

Networks which oscillate are of particular interest due to the critical role they play in regulating functions such as embryonic development and circadian rhythms. For example, nearly all life forms with bilateral symmetry are dependent on a process called segmentation in which oscillating waves of protein move through the developing embryo mapping out the locations of future limbs (Francois, Hakim and Siggia, Deriving Structure 2007). We are

interested in exploring whether this behavior can arise in a system that only contains the most basic reaction types so long as the selection criteria reward oscillating behavior.

1.1 Literature Review:

Implemented Models:

The following papers contain detailed descriptions of the networks that were chosen for replication during the first part of the project. Models were chosen for replication starting with those that were simple in terms of number of reaction types and models for which network diagrams and example behavior were provided. This allowed verification that the replication was accurate and all aspects of the model's implementation were correctly interpreted.

1. "A Case Study of Evolutionary Computation of Biochemical Adaptation" (Francois and Siggia, A Case Study 2008). The models in this paper are adaptive, that is, they respond to a change in the level of an input protein (I) with a sharp spike in the level of the observed protein (O), followed by quick adaptation back to a constant level. Three different network types were evolved and are described in detail. The first two models depend on protein - protein interactions to achieve adaptation, while the third model adds a transcription factor reaction, in which the input protein (I) can become a transcriptional regulator of the other proteins in the system. This paper also described the fitness function used to select networks in detail, allowing it to be reproduced. This paper also provides some plots showing how the network fitness changed over time until it converged on the desired behavior (Francois and Siggia, A Case Study 2008).
2. "Design of Genetic Networks with Specified Functions by Evolution in Silico" (Francois and Hakim, Design of Genetic Networks 2004). This paper describes two models. The

first is a bistable switch in which levels of a first protein (A) are high and levels of a second protein (B) are low, until an injection of protein B causes the system to switch to a state in which levels of B remain high, and levels of A drop and remain low. The second model is an oscillating network in which the level of an observed protein (O) alternates between very low and very high levels. The reactions and fitness functions used to evolve these two networks are described in detail, as are the initial network and general mutation procedure (Francois and Hakim, Design of Genetic Networks 2004).

Additional Papers:

In the following set of additional papers, other, sometimes more complicated models that have been developed are described. These provide examples of the application of in-silico evolution to a variety of systems and show how it can be used to answer questions and derive new models.

3. "Deriving Structure from Evolution: Metazoan Segmentation" (Francois, Hakim and Siggia, Deriving Structure 2007) describes an important application and extension that can be made from an oscillating network such as described in "Design of Genetic Networks with Specified Functions by Evolution in Silico" (Francois and Hakim 2004). This motivates the project goal of evolving an oscillating network, since it shows how such a network is the basis of embryonic segmentation. A key step in development of all metazoans involves the establishment of segments through repeated oscillating "waves" of protein travelling through a simple protein gradient in the embryo. These segments form the basis for all further development, providing biochemical signals that, for example, motivate one set of cells to develop into legs and another set of initially

identical cells to develop into torso (Francois, Hakim and Siggia, Deriving Structure 2007).

4. “Phenotypic Models of Evolution and Development: Geometry as Destiny” (Francois and Siggia, Phenotypic Models 2012). This is another more recent paper that describes the improvements in understanding of embryonic development that can be achieved by modelling it using an evolved network. Taking a fully functional organism and working backward results in a huge number of genes, proteins, and reactions that must be considered. Evolving a network can be used to arrive at a model with the key behavior, but vastly simplified (Francois and Siggia, Phenotypic Models 2012).
5. “Critical Timing Without a Timer for Embryonic Development” (Tufcea and Francois 2015). A basic oscillating network was described in “Design of Genetic Networks with Specified Functions by Evolution in Silico” (Francois and Hakim 2004), and then expanded to show how metazoan segmentation can be evolved in “Deriving Structure from Evolution: Metazoan Segmentation” (Francois, Hakim and Siggia 2007). This paper represents a further development by taking the evolved model and applying it to the key developmental protein Sonic Hedgehog. Again, this paper is of interest because it shows how the method explored in this project can be applied to a wide range of systems and yield valuable results. The authors recommend the model (arrived at through evolution in-silico) because it is simpler “more parsimonious” (Tufcea and Francois 2015, 1724) than other models currently available. This highlights a benefit of using the in-silico method (Tufcea and Francois 2015).

6. “Principles of Adaptive Sorting Revealed by In-Silico Evolution” (Lalanne and Francois 2013). This paper describes an application of the in-silico evolution method to study of the immune system, specifically, the process by which T-cells differential host cells from foreign cells. By applying the method, the authors were able to identify a key feature of all successful networks which they refer to as “adaptive sorting”. While none of the models from this paper are replicated in this project, it offers an example of how in-silico evolution can offer insight into a wide range of biochemical processes provided a suitable fitness function is provided (Lalanne and Francois 2013).
7. “Evolving Phenotypic Networks In-Silico” (Francois, Evolving Phenotypic Networks in Silico 2014) is one of the more recent papers reviewed and provides an overview of the methods used to model networks and their evolution, and its application to questions in biology and biochemistry. Three main types of models are described, adaptation, ligand discrimination, and segmentation (based on an oscillating model). Francois also discussed the critical role of the fitness function in successfully evolving a desired behavior (Francois, Evolving Phenotypic Networks in Silico 2014).

2 Review of In-Silico Evolution:

2.1 General Model:

Notation:

This first section describes the relationship between biochemical reactions in a network consisting of genes and proteins, the notations used to express these relationships, and how these are in turn expressed as terms in a system of ordinary differential equations (ODEs) that

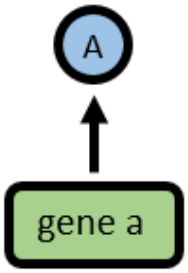
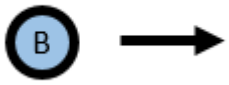
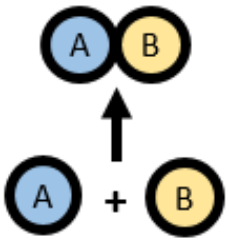
describe the network. In the equation for a given protein, each term comes from a reaction that protein is involved in so the solution of the equation yields the level of that protein in the system. These terms are based on mass action kinetics and depend on the concentrations of all proteins involved in the reaction and a kinetic constant. Proteins are identified by capital letters, for example A, and the concentration of a specific protein is denoted by enclosing the letter associated with the protein in square brackets, for example $[A]$. Generally, a kinetic constant is a small Greek letter with a subscript denoting the protein it is associated with, for example ρ_A. These conventions were noted throughout the papers and have been adhered to throughout this project. This system assists in identifying the roles of the various kinetic constants that appear in evolved networks (Francois and Hakim, Design of Genetic Networks 2004, 581), (Francois and Siggia, A Case Study 2008, 10)

Table 1. Naming scheme for kinetic constants based on (Francois and Hakim, Design of Genetic Networks 2004) and (Francois and Siggia, A Case Study 2008).

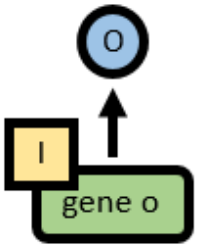
Greek letter denoting a kinetic constant	Associated reaction type
ρ	Rate of unregulated production
δ	Degradation rate
ϵ	Rate of formation of a protein complex
d	Rate of degradation of protein complex
τ	Delay in transcriptional regulation
γ	Rate of degradation of one protein or complex by another

The following table summarizes the reaction types seen in the networks giving the representation of the reaction in a network diagram, reaction type, and term that would be used to represent the reaction in the system of ODEs.

Table 2. Reaction types and corresponding term for differential equation. Based on the tables presented in the table on page 581 of "Design of Genetic Networks" (Francois and Hakim 2004) and in Appendix A on page 10 of "A Case Study" (Francois and Siggia 2008).

<table>
<tr><td>Network diagram representation</td><td>Term for differential equation</td></tr>
<tr><td colspan="2">Unregulated production</td></tr>
<tr><td> 
</td><td>$\frac{dA}{dt} = \rho_A$</td></tr>
<tr><td colspan="2">In this reaction type the protein is produced from the gene at a constant rate, ρ_A.</td></tr>
<tr><td colspan="2">Degradation of a protein or complex</td></tr>
<tr><td> 
</td><td>$\frac{dB}{dt} = -\delta_B$</td></tr>
<tr><td colspan="2">In this reaction the protein is degraded at a constant rate, δ_B. The amount of protein degraded is dependent on current concentration of the protein, so in the term for this reaction the kinetic constant is multiplied by the protein concentration.</td></tr>
<tr><td colspan="2">Formation of a protein complex</td></tr>
<tr><td> 
</td><td>$\frac{dA}{dt} = -\epsilon[A][B] + d[A][B]$
$\frac{dB}{dt} = -\epsilon[A][B] + d[A][B]$
$\frac{dAB}{dt} = \epsilon[A][B] - d[AB]$</td></tr>
<tr><td colspan="2">$\epsilon[A][B]$ is the rate of formation of the new protein complex AB, and $d[A][B]$ is the rate at which the complex dissociates back into proteins A and B. Note that the rate of formation is a negative term in the equations for proteins A and B, but positive term in the equation for the new protein complex.</td></tr>
</table>

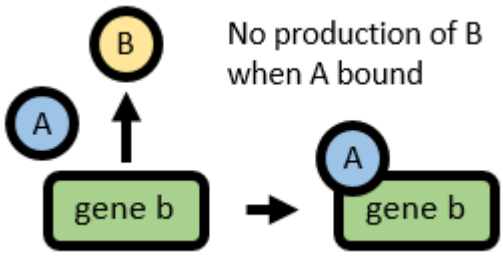
Transcriptional regulation of one protein by another



$$\frac{dO}{dt} = \frac{\rho_O}{\left(1 + \left(\frac{I(t-\tau_O)}{m}\right)^n\right)}$$

In this version of transcriptional regulation the production rate of the regulated protein, here O, is related to the level of the protein I by a Hill function. t is the current time in seconds, so $I(t-\tau_O)$ is the level of protein I τ_O seconds in the past. This models the delay between binding of a regulatory protein and the subsequent effect on production (Francois and Siggia, A Case Study 2008, 3).

Another representation of transcriptional regulation

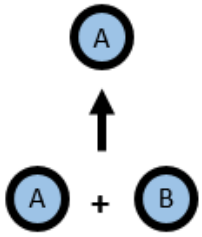


$$\frac{dB}{dt} = \rho_B\left[b_{(unbound)}\right]$$

$$\frac{db_{(bound)}}{dt} = binding\ rate\ \left[b_{(unbound)}\right][A]$$

$$\frac{db_{(unbound)}}{dt} = un - binding\ rate\ [b_{(bound)}]$$

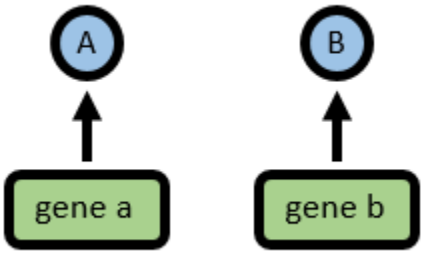
In this model of transcriptional regulation, the binding of protein A to gene b is modelled using mass action kinetics. The rate of production of protein B is then dependent on the ratio of gene b that is in the unbound state. Note that this system requires the levels of gene b to be explicitly tracked. The quantity of gene b is fixed, and is 1 in the paper (Francois and Hakim, Design of Genetic Networks 2004, 581).

Degradation of one protein by another	
	$\frac{dB}{dt} = -\gamma_{AB}[A][B]$
Note that this reaction contributes a negative term to the equation for the protein that is degraded, but doesn't affect levels of the other protein.	

Starting Network and Solving For Protein Levels over Time:

Each network is initialized with only two genes, and the proteins they produce. Each protein is assigned a unregulated production rate and a degradation rate. There are no other reactions (Francois and Hakim, Design of Genetic Networks 2004, 581).

Figure 1. Network diagram for starter network with just two genes and two proteins.



For example, in the starter network described above, there are two proteins (A and B) so we have two equations in the system. Each equation has a term for unregulated production which is constant (not affected by levels of regulatory proteins) and a degradation rate which depends on the level of the protein in the system and a fixed degradation rate (Francois and Hakim, Design of Genetic Networks 2004, 581).

Equation 1. System of ODEs for a starter network

$$\frac{dA}{dt} = \rho_A - \delta_A[A]$$

$$\frac{dB}{dt} = \rho_B - \delta_B[B]$$

For some protein X:

- ρ_X is the constant production rate associated with protein X
- δ_X is the constant degradation rate associated with protein X

Reaction types were determined based on the table of reaction types provided in Figure 2 of "Design of Genetic Networks with Specified Functions by Evolution in Silico" (Francois and Hakim 2004).

Solving these equations results in solution curves showing how the levels of protein in the system change with time. For the models shown in the paper, the authors used Euler's algorithm (Francois and Siggia, A Case Study 2008, 11) or the Runge-Kutta algorithm (Francois and Hakim, Design of Genetic Networks 2004, 581) for approximating the solutions to ODEs.

The Fitness Function:

The fitness function takes one protein out of the system (sometimes called the "observed" protein) and assigns the network a score based on how closely the levels of that protein over time approach the desired behavior profile. This is achieved by running the fitness function on the solution curve corresponding to the observed protein that was found while solving the system of ODEs. An important aspect of the fitness function is that it must be able to assign marginally better scores as networks show marginally better, but not yet ideal behavior. This allows selection of gradually better and better networks until the desired

behavior is achieved. In many models, the score assigned by the fitness function approaches zero as network behavior improves, so the goal of selection is to arrive at the lowest possible score (Francois and Siggia, A Case Study 2008, 3).

Mutation and Selection:

A set of networks is initialized. This set usually consists of 100 networks (Francois and Hakim, Design of Genetic Networks 2004, 580). As described above under starting networks, each of these networks contains only two genes and the proteins they produce. In each round of mutation, the system of ODEs that represent the network is solved over a fixed time period, and the solution corresponding to the observed protein is passed to the fitness function for scoring. After all networks have been assigned a score, the 50 networks with the best scores (if 0 corresponds to ideal behavior, then it would be the 50 lowest scoring networks) are retained, while the other 50 are deleted. This is the selection process (Francois and Hakim, Design of Genetic Networks 2004, 580). Then, each of the 50 surviving networks undergoes two mutations. The basic mutation types are described in "Design of Genetic Networks with Specified Functions by Evolution In Silico" (Francois and Hakim 2004, 581):

A. A protein can be added to the network
B. An interaction between two proteins (or protein complexes) can be added to form a new protein complex
C. A protein degradation rate can change
D. Any other kinetic constant can change

E. A gene or interaction can become “broken”, causing the rate of production of a protein or complex to go to zero
F. Which protein is “observed” by being passed to the fitness function can change
G. One protein can become a transcriptional regulator of another

This list covers the basic mutation types. More complicated networks can contain additional reactions such as a reaction in which one protein degrades another protein or phosphorylation of one protein by another (Francois and Hakim, Design of Genetic Networks 2004). All models reviewed contained at least mutations A through E. After that, the inclusion/exclusion of specific reactions that can be added through mutation seems to be guided by the type of reactions seen in the in-vivo examples of the given network type since it is often desired that the evolved in-silico model resemble the living system (Francois and Siggia, A Case Study 2008) (Francois, Hakim and Siggia, Deriving Structure 2007). Which mutation to perform on each round is selected randomly from the available mutations types with a uniform probability distribution. This leaves 50 surviving networks, and 50 mutated networks for another set of 100 networks, and the whole process is completed. This can be continued until the desired level of fitness is reached (Francois and Hakim, Design of Genetic Networks 2004, 580).

2.2 Software Implementation:

For this project, the networks were modelled using Python 3.5 installed with Anaconda. Programs were written in the Spyder IDE, and notes were kept on progress, problems, and intermediate steps of the project using Jupyter notebooks.

Program:

A goal of this project was to independently replicate models described in the papers, and also create a program capable of achieving a functional network through mutation.

The basic network structure was implemented using a Python class object containing:

A. A system of ODEs

B. A set of reaction constants

Class Methods:

I. Solve itself using the Runge-Kutta algorithm to numerically solve ODEs

II. Score itself according to the provided fitness function

III. Select a mutation method from a set with uniform probability, and apply that mutation to itself

IV. Mutation methods for each of the mutation types described above under Mutation and Selection

V. Methods corresponding to each of the possible reaction types in the system:

    a. Unregulated production

    b. Degradation

    c. Interaction to form another protein

    d. Being formed from an interaction

    e. Transcriptional regulation

A challenge in creating this program was how to represent the network structure inside a class without creating a lot of needless objects. The solution was to allow a specific network to be effectively defined by its system of ODEs and dictionary of kinetic constants. In the program, each ODE is represented as a list of reactions, with each reaction constituting one

term in the equation. These reactions are specified by a string which corresponds to the class method for that reaction and a list of arguments. These arguments are also strings, each of which is a key to a dictionary containing every kinetic constant for the network. This structure allows the ODE to be modified (mutated) by editing the list of strings and changing the values in the arguments dictionary. Then, when the ODE is needed for solving the system, it can be compiled by converting all the arguments into values using the dictionary, and giving these to the reaction method corresponding to the method string.

To return to the example of the simple starter network, the first equation in the system of ODEs:

$$\frac{dA}{dt} = \rho_A - \delta_A[A]$$

Would be represented as:

['unregulated_production', ['rho_A', 'delta_A'], 'degradation', ['A_level', 'delta_A']]

A concern with this approach to representing the network is that it might not be the most efficient, which could become a problem when running simulations with a large number of networks for multiple rounds of mutation and selection. However, a benefit is that all the strings make the network relatively human-readable. After mutation, the resulting network can be deciphered and represented as a network diagram or written out as a system of ODEs with constants. In order to address concerns about accuracy and speed of running, a fourth order Runge-Kutta algorithm was used instead of Euler's algorithm.

3 Results

3.1 Reproducing Example Networks from the Papers:

For the first stage of the project, the goal was to replicate several networks described in the papers. This was done by determining the system of equations and constants associated with the network, and hardcoding these into the network class. Then, the network behavior could be assessed visually by running the Runge-Kutta method on it and plotting the resulting solutions, then comparing this to plots provided in the papers.

Adaptive Networks

Three basic network types that were produced by in-silico mutation are provided in the paper "A Case Study of Evolutionary Computation of Biochemical Adaptation" (Francois and Siggia 2008). These networks respond to changes in the level of an input protein with a sharp spike, followed by a rapid adaptation and return to a steady level. This is a critical characteristic in any system that needs to respond strongly to a change in the level of some indicator, but then return to normal when hat indicator is not changing. The paper shows that three very different networks can be arrived at through mutation, all of which manage to produce this behavior type. In this project, the first and simplest network, described as a buffered output system, and the third network, a system with transcriptional regulation are reproduced (Francois and Siggia, A Case Study 2008). In the following section, the equations and network diagrams of these two networks are described, and the plots produced by replication are given.

Buffered Output Adapting Network

In the first system the observed protein (O) is produced at a constant rate. It undergoes a protein-protein interaction with the injected protein (I) to produce protein C (Francois and Siggia, A Case Study 2008, 4).

This network type is described by the general system of ODEs:

Equation 2. Generalized Equations for a buffered output adaptive system (Francois and Siggia, A Case Study 2008, 4).

$$\frac{dO}{dt} = \rho - \epsilon[O][I] + d[C] - \delta_O[O]$$

$$\frac{dC}{dt} = \epsilon[O][I] - d[C] - \delta_C[C]$$

Figure 2. Network diagram for a buffered output adaptive system (Francois and Siggia, A Case Study 2008, 4).

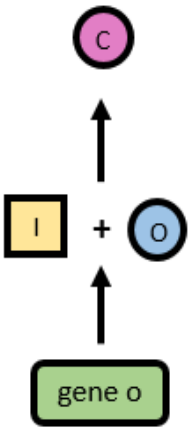
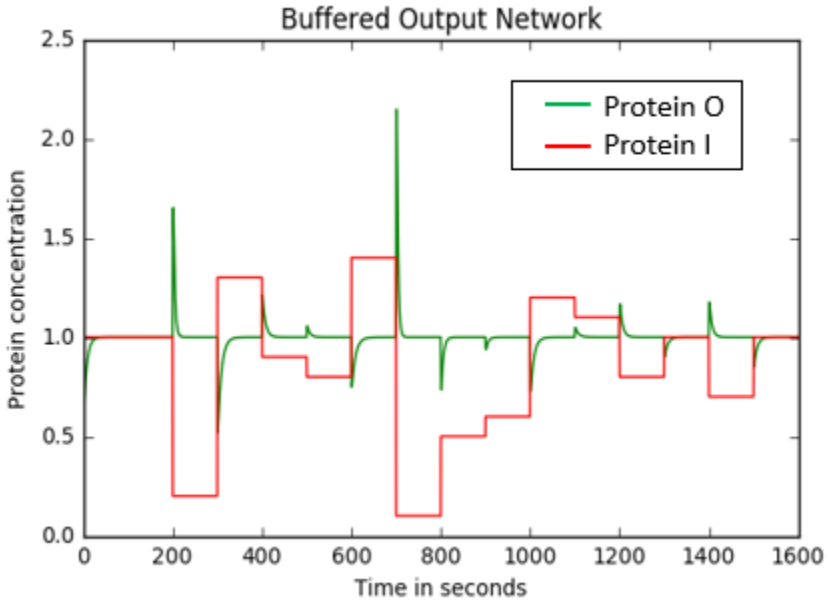


Table 3. Constants for the example buffered output network (Francois and Siggia, A Case Study 2008, 4).

$\rho = 1$
$\epsilon = 2$
$\delta_O = 1$
$d = 0.5$
$\delta_C = 0$

Combining these constants with the system of ODEs yielded a network system which could be hard-coded into the network class. Solving it with the Runge - Kutta algorithm produces the following behavior, which matches the behavior shown in Figure 3 of the paper (Francois and Siggia, A Case Study 2008, 4):

Figure 3. Levels of the input (I) and observed (O) proteins over time in the buffered output adapting network.



Transcription Factor Network

The next network to be replicated is the transcription factor network. In the previous network the desired behavior was achieved through evolution without providing a transcriptional regulation reaction. In this third network, a new reaction is introduced in which the input protein (I) can bind to genes and either promote or repress production of the protein associated with that gene (Francois and Siggia, A Case Study 2008, 3).

This interaction between I and the production rate of other system proteins is modelled using a Hill equation (Francois and Siggia, A Case Study 2008, 3):

Equation 3. Example Hill function modelling transcription factor binding to a gene (Francois and Siggia, A Case Study 2008, 10).

$$\frac{dX}{dt} = \frac{\rho_X}{1 + \left(\frac{I(t - \tau_X)}{X_0}\right)^n}$$

The expression $I(t - \tau_X)$ enforces a slight delay between a change in the level of I and a change in the rate of production for protein X by having the production term for X reference the level of I at $(t - \tau_X)$ seconds, with t being the current time in seconds (Francois and Siggia, A Case Study 2008, 6). The networks evolved once transcriptional regulation was introduced are modelled by a three protein system of ODEs.

Equation 4. System of ODEs for the adaptive network with transcriptional regulation (Francois and Siggia, A Case Study 2008, 6).

$$\frac{dO}{dt} = f\big(I(t - \tau_O)\big) - \epsilon[R][O] + d[C] - \delta_O[O]$$

$$\frac{dR}{dt} = g\big(I(t - \tau_O)\big) - \epsilon[R][O] + d[C] - \delta_R[R]$$

$$\frac{dC}{dt} = \epsilon[R][O] - d[C] - \delta_C[C]$$

The network diagram associated with this system of ODEs is provided in Figure 7 of the paper. In this network the production of both the observed protein (O) and another protein (R) is regulated by the input protein (I) binding to the regulatory regions for the genes associated with O and R. O and R then undergo a protein - protein interaction to produce a fourth protein, C (Francois and Siggia, A Case Study 2008, 7).

Figure 4. Network diagram for an adaptive system in with transcriptional regulation by the input protein (I) (Francois and Siggia, A Case Study 2008, 7).

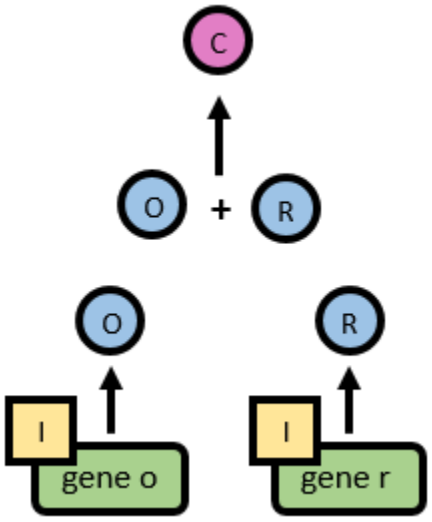
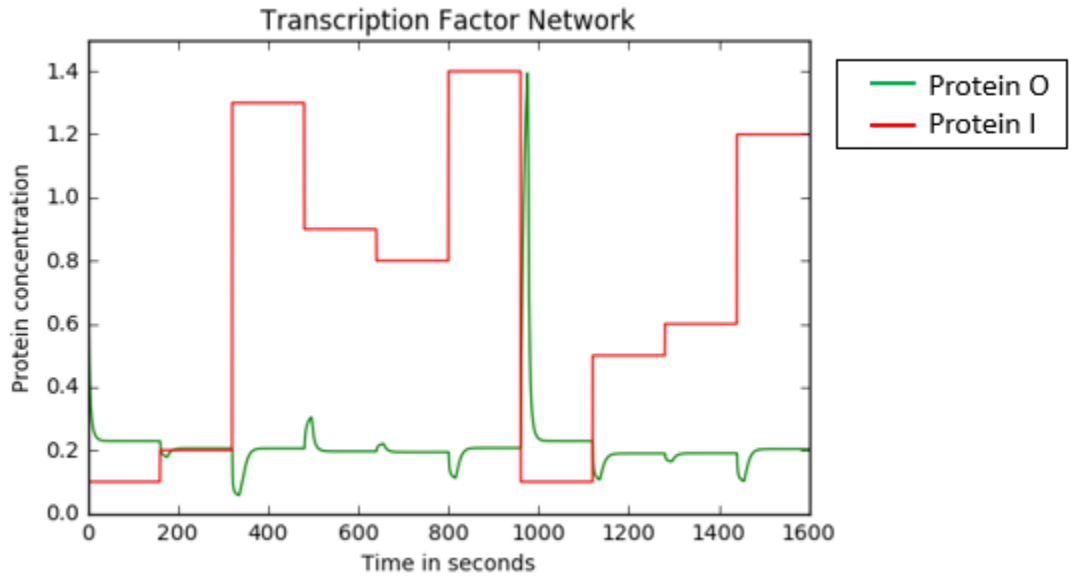


Table 4. Reaction constants for the example transcription factor network (Francois and Siggia, A Case Study 2008, 8).

$f(I) = \dfrac{0.6}{1 + \left(\dfrac{I(t - \tau_O)}{0.54}\right)^1}$
$g(I) = \dfrac{0.96}{1 + \left(\dfrac{I(t - \tau_R)}{0.7}\right)^{1.4}}$
$\delta_O = 0.18$
$\delta_R = 0.14$
$\epsilon = 0.9$
$d = 0.3$
$\delta_C = 0.8$
$\tau_O = 0.8$
$\tau_R = 16$

Figure 5. Levels of the input (I) and observed (O) proteins over time in the transcription factor adapting network. This behavior matches what is seen in the example network in the paper (Francois and Siggia, A Case Study 2008, 8).

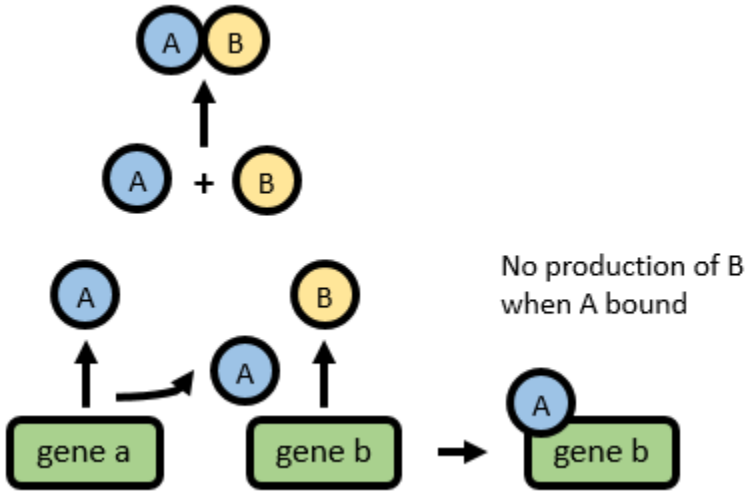


Bistable Switch Network

The final network to be replicated through hardcoding a specific example given in a paper is the bistable switch network from "Design of Genetic Networks with Sepcified Function by Evolution in Silico" (Francois and Hakim 2004). This network has two main proteins, A and B. The desired behavior is that levels of A will remain high until a large injection of protein B is delivered. Once this occurs, the levels of A will drop and the levels of B will remain high. This is an important characteristic for any system that needs to alternate between being fully in one state or fully in another state (Francois and Hakim, Design of Genetic Networks 2004, 582). Several networks are presented, for this project the network presented in Figure 3 on page 583 was chosen. For this network, the authors provide a list of all reactions in the system, all constants, a network diagram, and a plot showing typical behavior of the network. In order to

replicate it, it was necessary to go through these reactions and translate them into terms for a system of ODEs.

Figure 6. Network diagram for an example bistable switch system (Francois and Hakim, Design of Genetic Networks 2004, 583).



From examining this network diagram, it is possible to see how two distinct states are achieved. When levels of A are high, protein A binds to gene b and represses it. This leads to a high level of protein A in the system with very little going into the dimerization reaction with protein B since there is so little protein B. As soon as levels of protein B increase (due to a large influx of B into the system), all protein A is taken up by the dimerization reaction with B, and none remains to repress gene b. Since gene b is no longer repressed, even more protein B enters the system and it continues to take up all the protein A produced (Francois and Hakim, Design of Genetic Networks 2004, 583).

Figure 7. State 1: High levels of protein A, low levels of protein B (Francois and Hakim, Design of Genetic Networks 2004, 583).

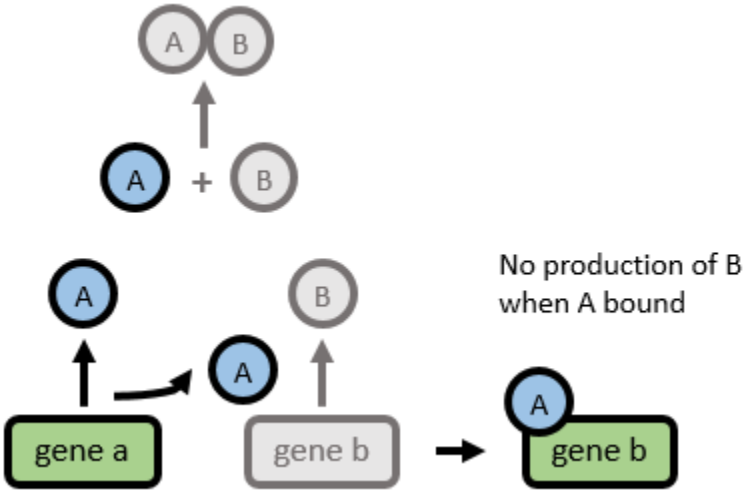
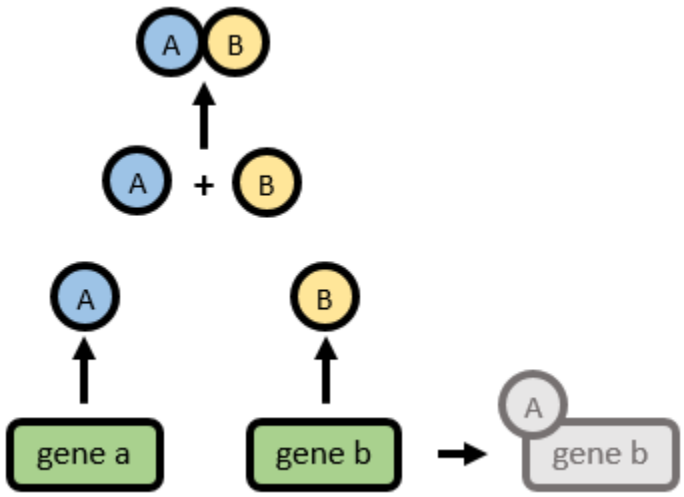
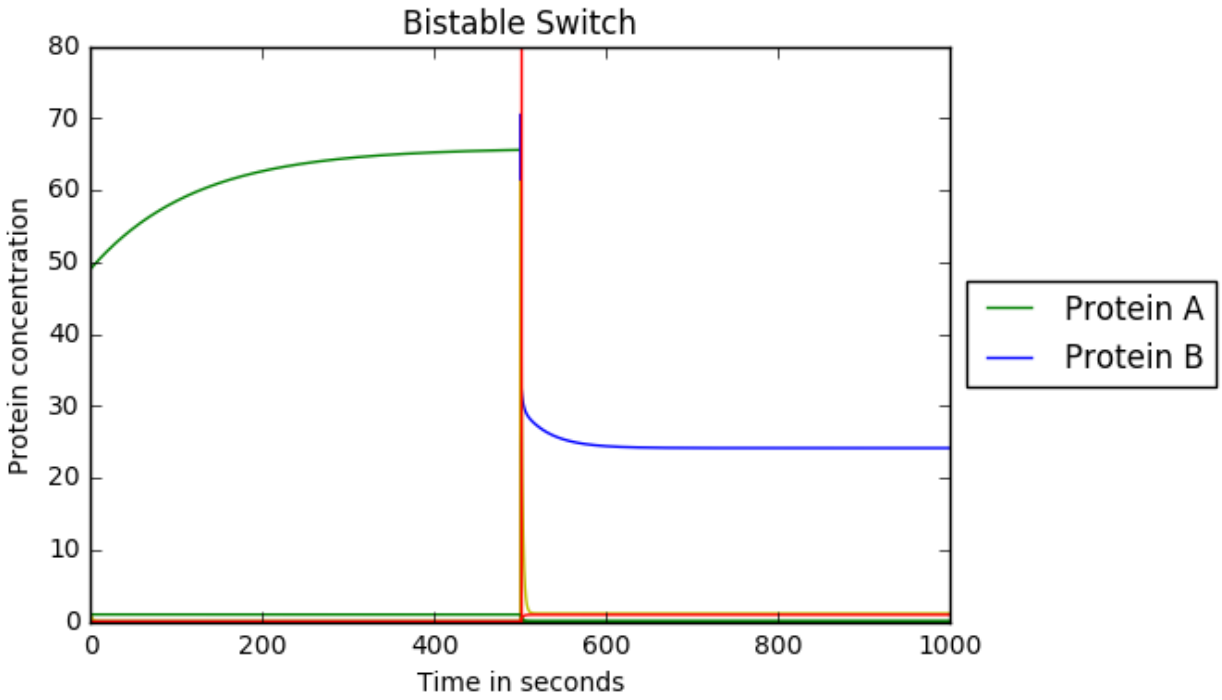


Figure 8. State 2: High levels of protein B, low levels of protein A. All protein A produced immediately undergoes dimerization with protein B (Francois and Hakim, Design of Genetic Networks 2004, 583).



Then, using the set of reactions and their constants provided with the network diagram, we can construct a system of ODEs to describe this system term by term. After hardcoding this system of differential equations into a network, the following plot was obtained.

Figure 9. Bistable network. We can see that following the injection, the level of B increases and the level of A goes to nearly zero.



3.2 Mutating Networks to Gain Functionality

Once several specific networks had been successfully replicated, the next goal of this project was to successfully mutate a new network from scratch.

Adapting Network with Basic Reactions

The first mutation was done using the set of possible reaction types seen in the first two adapting networks from "A case study of evolutionary computation of biochemical adaptation" (Francois and Siggia 2008)

A. A protein can be added to the network
B. An interaction between two proteins (or protein complexes) can be added to form a new protein complex
C. A protein degradation rate can change

D. Any other kinetic constant can change
E. A gene or interaction can become “broken”, causing the rate of production of a protein or complex to go to zero
F. Which protein is “observed” by being passed to the fitness function can change

In order to score the networks, a fitness function had to be developed that would reward spiking followed by adaptation in response to changes in the levels of the input protein (I). The fitness function used to evolve the example adaptive networks is described in some detail. Key elements of the fitness function are provided in Equations 1 and 2 on page 2 (Francois and Siggia, A Case Study 2008).

Equation 5. Key components of the fitness function for an adaptive network (Francois and Siggia, A Case Study 2008, 2).

$$\Delta O_{ss} = max|O(I_2) - O(I_1)|$$

$$\Delta O_{max} = max|O(t) - O(I_1)|$$

$$O_{avg} = mean(O\ levels)$$

ΔO_{ss} gets the difference between the levels of the observed protein O at two different levels of the input protein, I_1 and I_2. A key feature of adaptation is that O goes back to a consistent level when I is not changing, so this term is used to penalize networks that do not return to a consistent level. ΔO_{max} gets the maximum difference between the level of the observed protein at time t following a change in the level I, and the level of O while I remains fixed at level I_1. This term rewards high spikes in response to changes in levels of the injected protein (I) (Francois and Siggia, A Case Study 2008, 2). The final term incorporated in the fitness function is the average levels of protein O over the full period, O_{avg}. This term is then

multiplied by a constant ϵ which is small compared to typical levels of protein O (Francois and Siggia, A Case Study 2008, 3). These terms are combined to get the full fitness function.

Equation 6. Equation for the fitness score, based on descriptions in the paper and used to evolve an adaptive network.

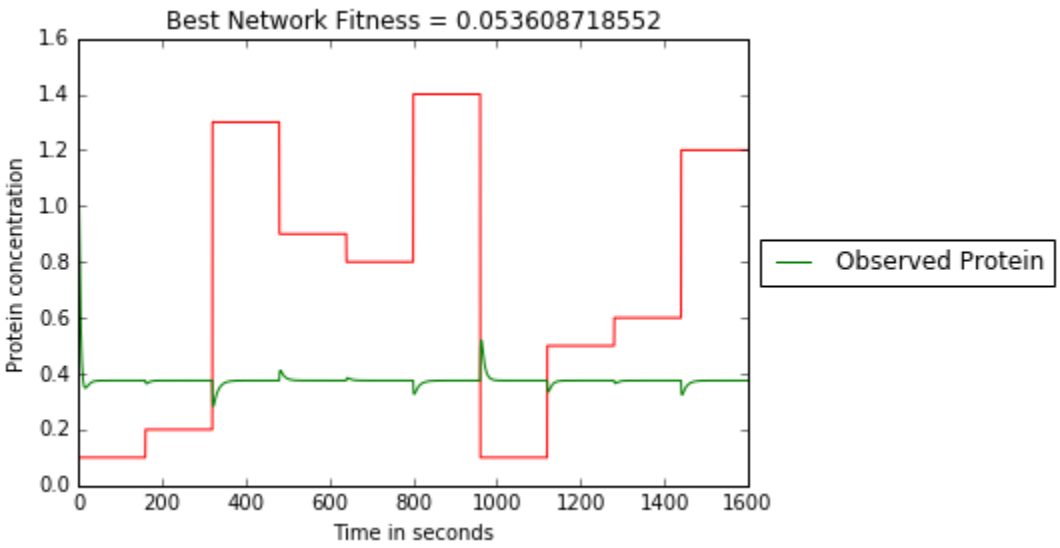
$$fitness\ score = \frac{\Delta O_{ss} + \epsilon O_{avg}}{\Delta O_{max} + 0.000001}$$

By combining the terms in this way, the fitness function gets smaller as the network behavior approached the desired behavior. As adaptation improves, the dominant term in the numerator, ΔO_{ss}, decreases. As spikes become more pronounced, the dominant term in the denominator ΔO_{max} increases. The purpose of the third term, ϵO_{avg} is to introduce a small, positive quantity into the numerator so it never goes all the way to zero. If this was not there, networks in which the observed protein flatlines and displays no spiking at all could get a perfect score (Francois and Siggia, A Case Study 2008, 3). Likewise, adding 0.000001 to the denominator prevents a fitness score of infinity being assigned when there is no spiking at all.

A final feature of the fitness function which was added in this project is to assign a score of infinity to any networks that began to show extreme oscillations, eventually going out of control. This out of control behavior cannot be handled by the Runge-Kutta algorithm, and would definitely constitute behavior that is not biologically desirable.

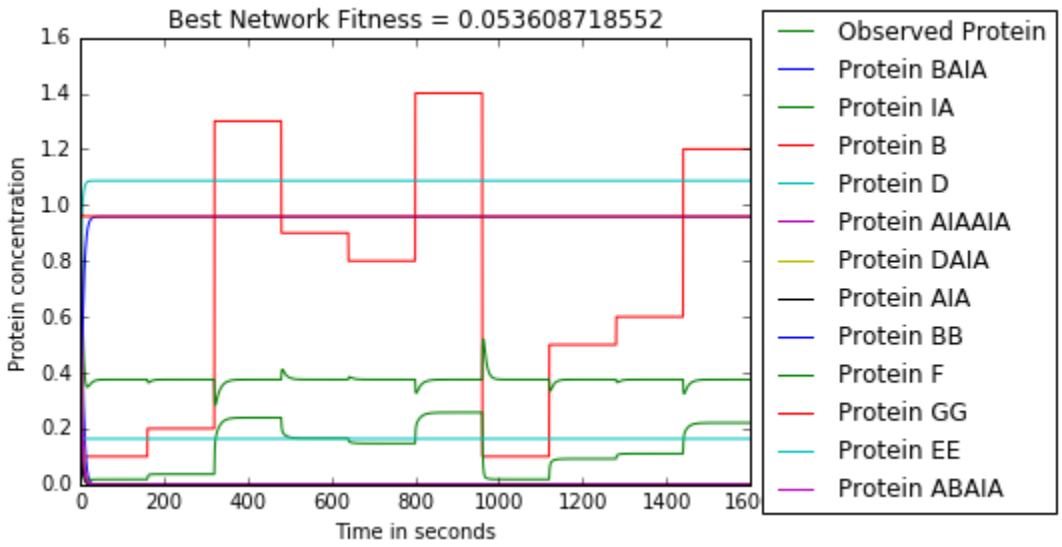
After implementing the fitness function, a set of ten starter networks (two genes, no interactions) was initiated and put through 30 rounds of mutation and selection. At the end of this, the network with the best fitness score was plotted.

Figure 10. Best network out of 10 following 30 rounds of selection and mutation, showing only the observed protein.



This can be compared to the behavior of the simple buffered network replicated in the earlier part of the project. There are clear spikes in response to changes in levels of the input protein I, followed by adaptation back to a steady level. This shows the even with a small number of networks and only 30 rounds of mutation, it is possible to achieve the desired behavior profile. To learn more about how this behavior is achieved, the levels of every protein in the system can be plotted.

Figure 11. Best network out of 10 following 30 rounds of selection and mutation, showing all proteins in the system.



Some proteins react to changes in the level of the input protein (I), but some others simply sit in the system and do not interact with anything. In order to produce the network diagrams given in the papers, proteins such as these were removed since they do not play any role in the behavior of interest. In order to see what reactions do underlie the behavior, we can look at the system of ODE terms and reaction constants saved in the network object.

Table 5. Reaction constants for evolved adapting network.

Production rates:
'A_rate': 0.06016240270307727
'B_rate': 0.925766640603965
'D_rate': 0.895507807110831
'EE_rate': 0.13332195236937117
'F_rate': 0.6651194720294824
'GG_rate': 0.6133885739898531
Complex breakdown rates:
'd_ABAIA': 0.6764631285796429

'd_AIA': 0
'd_AIAAIA': 0.2253476549483895
'd_BAIA': 0.022766836567611914
'd_BB': 0.8097582395582954
'd_DAIA': 0.03414758242595295
'd_IA': 0.6451557094562537
Degradation rates:
'delta_A': 0.16026160978539708
'delta_ABAIA': 0.6811896583880871
'delta_AIA': 0.06946940600311657
'delta_AIAAIA': 0.049058044315565374
'delta_B': 0.06448176319911547
'delta_BAIA': 0.2336045972491666
'delta_BB': 0.3650398574879512
'delta_D': 0.8238621262610408
'delta_DAIA': 0.5000175657628798
'delta_EE': 0.8147132259387762
'delta_F': 0.06081785380987903
'delta_GG': 0.6394813999932816
'delta_IA': 0
Complex formation rates:
'epsilon_ABAIA': 0.7242690836787159
'epsilon_AIA': 0.0
'epsilon_AIAAIA': 0.6207564549340701
'epsilon_BAIA': 0.6164281197017958
'epsilon_BB': 0.09120853497619241
'epsilon_DAIA': 0.886168335627356
'epsilon_IA': 0.31592633017888483

Equation 7. System of equations for the evolved adapting network

$$\frac{dA}{dt} = \rho_A - \epsilon_{IA}[A][I] + d_{IA}[IA] - \epsilon_{AIA}[A][IA] + d_{AIA}[AIA] - \epsilon_{ABAIA}[A][BAIA] + d_{ABAIA}[ABAIA] - \delta_A[A]$$

$$\frac{dBAIA}{dt} = \epsilon_{BAIA}[B][AIA] - d_{BAIA}[BAIA] - \epsilon_{ABAIA}[BAIA][A] + d_{ABAIA}[ABAIA] - \delta_B AIA[BAIA]$$

$$\frac{dIA}{dt} = \epsilon_{IA}[I][A] - d_{IA}[IA] - \epsilon_{AIA}[IA][A] + d_{AIA}[AIA] - \delta_{IA}[IA]$$

$$\frac{dB}{dt} = \rho_B - \epsilon_{BB}[B][B] + d_B B[BB] - \epsilon_{BAIA}[B][AIA] + d_{BAIA}[BAIA] - \delta_B[B]$$

$$\frac{dD}{dt} = \rho_D - \epsilon_{DAIA}[D][AIA] + d_{DAIA}[DAIA] - \delta_D[D]$$

$$\frac{dAIAAIA}{dt} = \epsilon_{AIAAIA}[AIA][AIA] - d_{AIAAIA}[AIAAIA] - \delta_{AIAAIA}[AIAAIA]$$

$$\frac{dDAIA}{dt} = \epsilon_{DAIA}[D][AIA] - d_{DAIA}[DAIA] - \delta_{DAIA}[DAIA]$$

$$\frac{dAIA}{dt} = \epsilon_{AIA}[A][IA] - d_{AIA}[AIA] - \epsilon_{AIAAIA}[AIA][AIA] + d_{AIAAIA}[AIAAIA] - \epsilon_{DAIA}[AIA][D] + d_{DAIA}[DAIA] - \epsilon_{BAIA}[AIA][B] - d_{BAIA}[BAIA] - \delta_{AIA}[AIA]$$

$$\frac{dBB}{dt} = epsilon_{BB}[B][B] - d_{BB}[BB] - \delta_{BB}[BB]$$

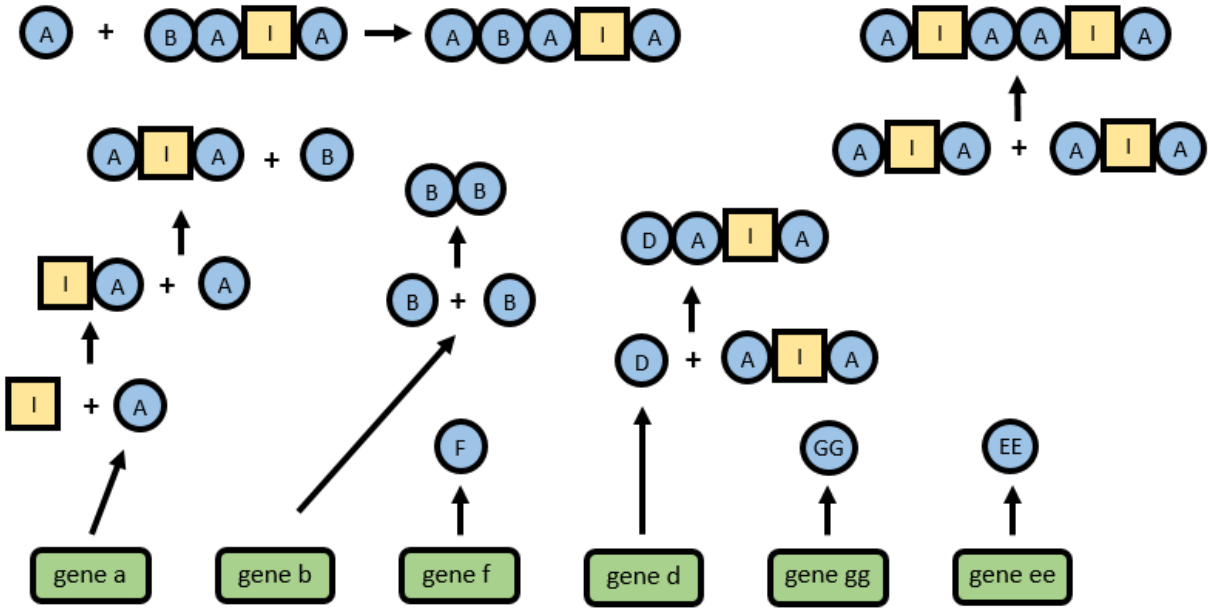
$$\frac{dF}{dt} = \rho_F - \delta_F[F]$$

$$\frac{dGG}{dt} = \rho_{GG} - delta_{GG}[GG]$$

$$\frac{dEE}{dt} = \rho_{EE} - \delta_{EE}[EE]$$

$$\frac{dABAIA}{dt} = \epsilon_{ABAIA}[A][BAIA] - d_{ABAIA}[ABAIA] - \delta_{ABAIA}[ABAIA]$$

Figure 12. Network diagram for the evolved adapting network. This can be determined by examining the lists of reactions saved in the final Network class object.



From the network diagram, it can be seen that reactions between the observed protein (A) and the input protein (I) were strongly favored.

A critical feature to get mutation that guides a network to desired behavior is that the fitness function rewards incremental improvements in behavior rather than rewarding only perfect behavior (Francois and Siggia, A Case Study 2008, 2). In order to assess this, at each round of mutation the best and mean fitness scores among the set of networks were found and saved. After mutation and selection was complete, these can be plotted to see how the fitness function score changed over generations of networks.

Figure 13. Best and mean fitness scores over 30 rounds of mutations.

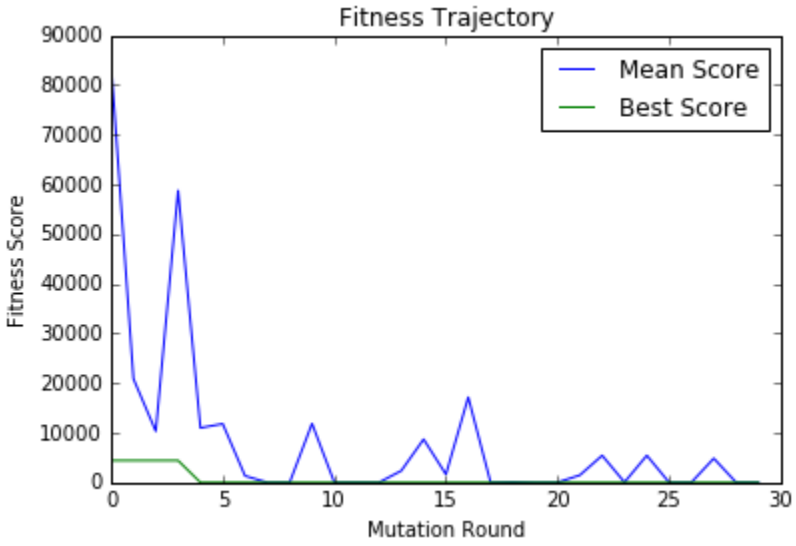
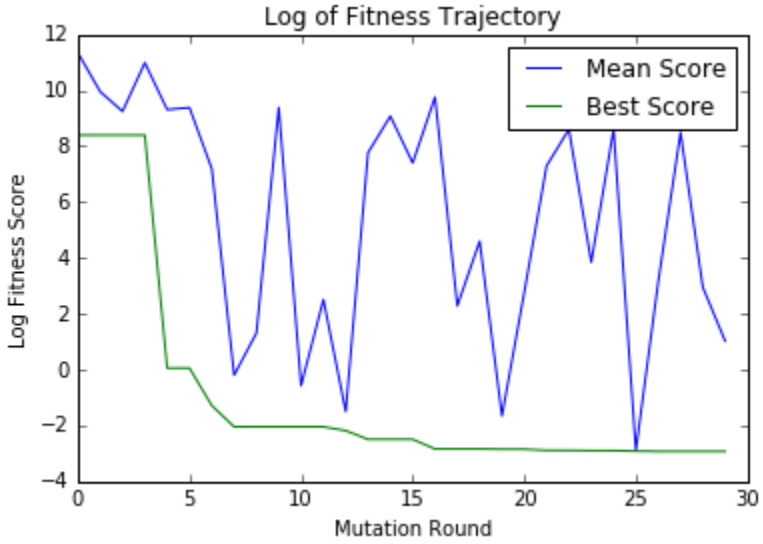


Figure 14. Best fitness scores dropped dramatically just prior generation five. To see this better, take the log of both best and mean fitness scores.

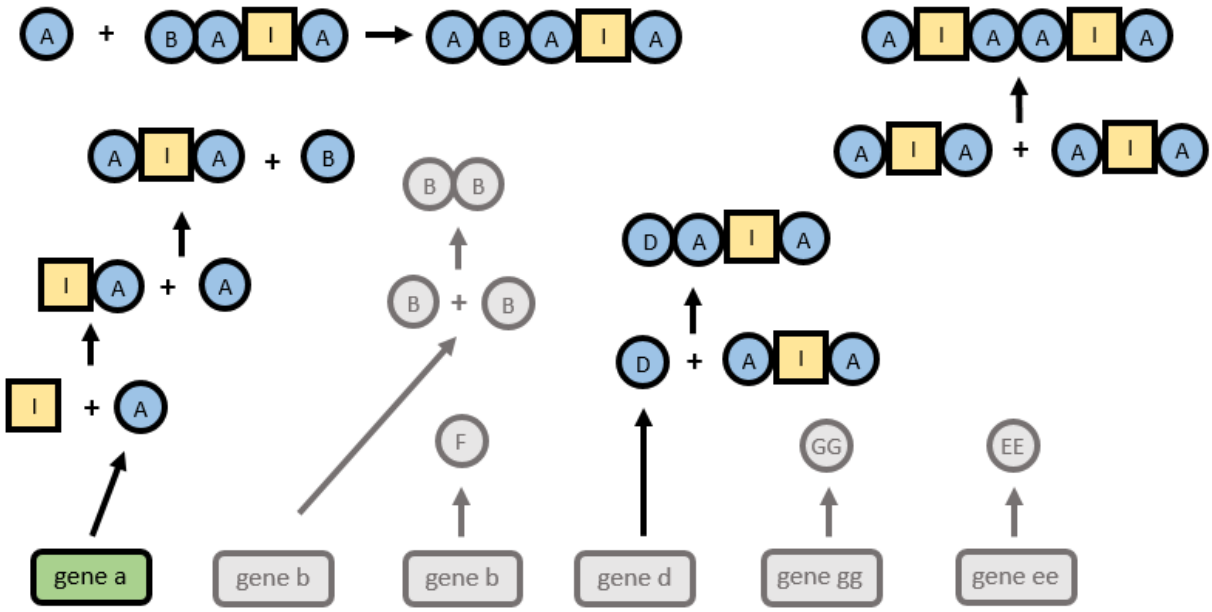


There was a good deal of bouncing around in the mean score. This is likely do to detrimental mutations that appeared in one round and that were then immediately selected against. The best score drops quickly just before the fifth round of mutation. A sudden, steep drop like this is also seen in a similar plot provided in Figure 7 in “A case study of evolutionary computation of biochemical adaptation”. The authors observe that the abrupt drop occurs

when a relationship between the input protein and the observed protein develops, indicating that this is a critical component of getting the desired behavior. It appears that the scaling of the fitness function developed for this project is different than that used by the authors, but it appears both versions are capable of rewarding desirable behavior (Francois and Siggia, A Case Study 2008, 7).

In the study of evolving adapting networks, the authors identify two different basic network types that can be produced with the same basic set of mutations and fitness function as were used to evolve the above network. This leads to the question of whether the new evolved network falls into one of these two types, and which one. There are several proteins that have no reactions with the observed protein, so these can be eliminated from the network. From inspecting the remaining reactions, it appears that the key factor in causing the observed protein (A in this network) to respond to changes in levels of the input protein I is a protein - protein interaction between A and I. Levels of A then adjust through the formation of additional protein complexes involving the proteins B and D. It is very likely that not quite so many interactions are required. The fundamental type here is buffered output (Francois and Siggia, A Case Study 2008, 4), the same type as the first network that was replicated earlier in this project.

Figure 15. Network diagram for the evolved adapting network with side reactions greyed out. They key reactions in getting the desired behavior are the formations of complexes with the proteins A and I.



Evolving an Oscillating Network

To address the third goal of this honors thesis, we take the program used to evolve the above network and substitute a fitness function that rewards oscillating behavior rather than adaptive behavior. The fitness function for an oscillating network is described on page 583 of "Design of genetic networks with specified functions by evolution in silico" (Francois and Hakim 2004). This network relies on transcriptional regulation. By attempting to evolve a network with similar behavior, but with only a very basic set of reactions that do not include transcriptional regulation, we can gain insight into whether transcriptional regulation is truly necessary for oscillations to set up. In the paper, Francois and Hakim report observing evolved networks that did not depend on transcriptional regulation for the final behavior (Francois and Hakim, Design of Genetic Networks 2004, 583).

The authors found that a very simple fitness function was the most effective at getting oscillations. This fitness function is defined by setting an upper and lower level for the observed protein. The period over which the protein levels are monitored is then divided into periods. The network is assigned a better fitness score if the levels of the observed protein approach the upper level at full periods, and the lower level at half periods (Francois and Hakim, Design of Genetic Networks 2004, 583). In order to implement this, an upper level, lower level, and period in seconds were set inside the fitness_function method of the network class. The calculation was kept very simple. For each full period, the level of the observed protein present at that time according to the Runge-Kutta algorithm was compared to the upper limit by getting the absolute value of the difference. These differences were added up for all full periods. At the half periods, the difference between the level of observed protein and the lower level was taken. The mutation and selection program then selected for the smallest score. This would be assigned to the network with the highest levels of observed protein at full periods and the lowest levels at half periods.

Equation 8. Fitness function for an oscillating network.

$$fitness\ score = |upper\ level - O_{2\pi k}| + |lower\ level - O_{\pi k}|$$

A set of 10 networks was put through 30 rounds of mutation. The upper level was set to 2, and the lower level was set to 0.5. The period was 100 seconds.

Figure 16. Behavior of observed protein in an evolved network using the oscillating fitness function.

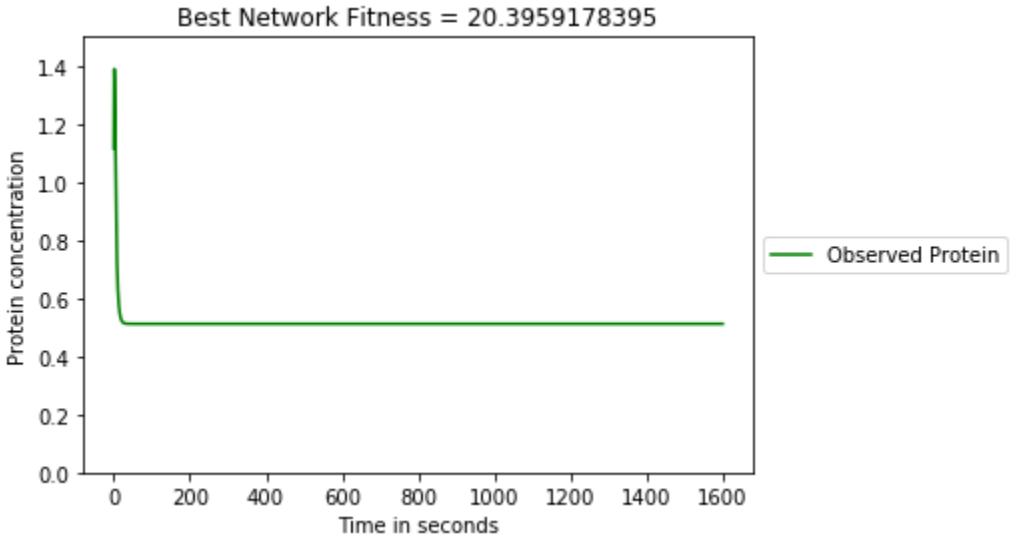


Figure 17. Behavior of all proteins in an evolved network using the oscillating fitness function.

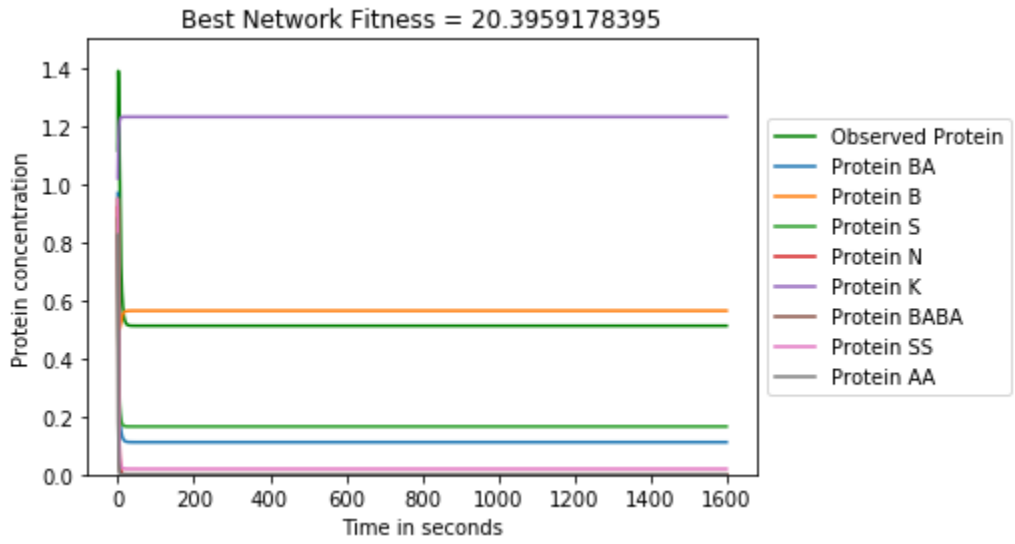
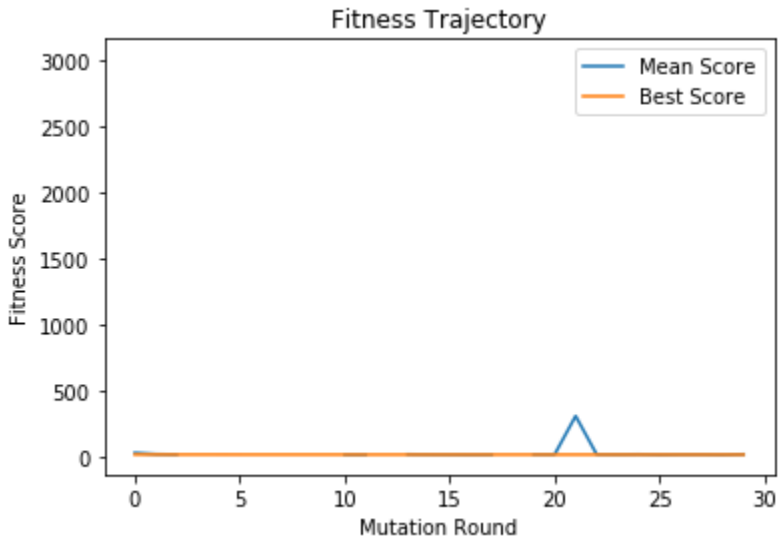


Figure 18. Trajectory of fitness scores in 30 rounds of mutation with the oscillating fitness function.



This procedure was repeated with 20 networks and 300 rounds of mutation, but no development of oscillations was seen. One possible source of the failure to mutate was an insufficiently sensitive fitness function that only rewards perfect behavior and does not respond to behavior that is better, but not yet optimal. To address this, a modified fitness function tried in which the sum of the distances from the desired levels is divided by the difference between the level at the half and full period. This rewards changing levels, and penalizes flatlining.

Equation 9. Modified oscillator fitness function that penalizes flatlining.

$$fitness\ score = \frac{|upper\ level - \ O_{2\pi k}| + |lower\ level - \ O_{\pi k}|}{(O_{2\pi k} - O_{\pi k}) * 10}$$

Unfortunately, this also was not sufficient to produce oscillations.

Figure 19. Observed protein from system evolved with the modified oscillator fitness function.

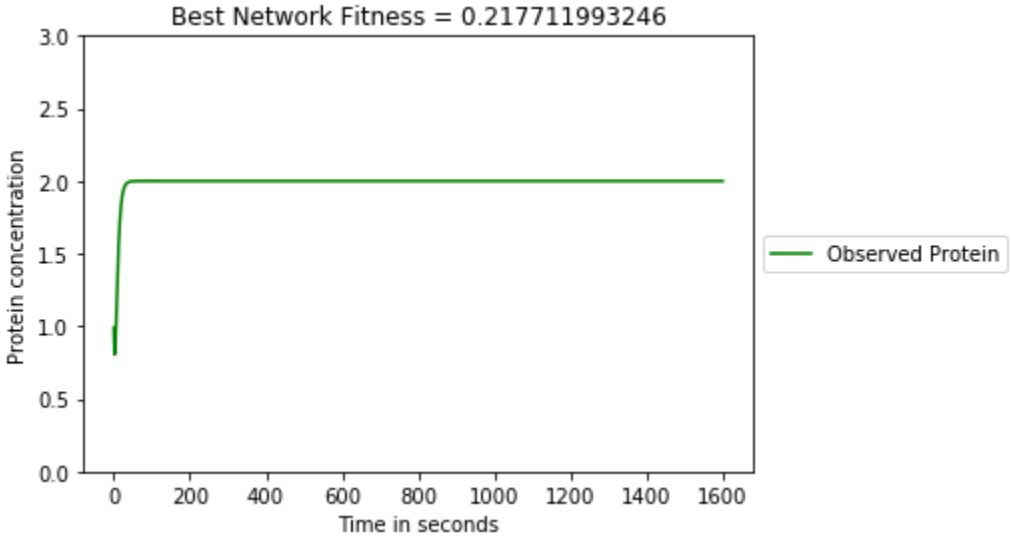


Figure 20. All proteins in system evolved with the modified oscillator fitness function.

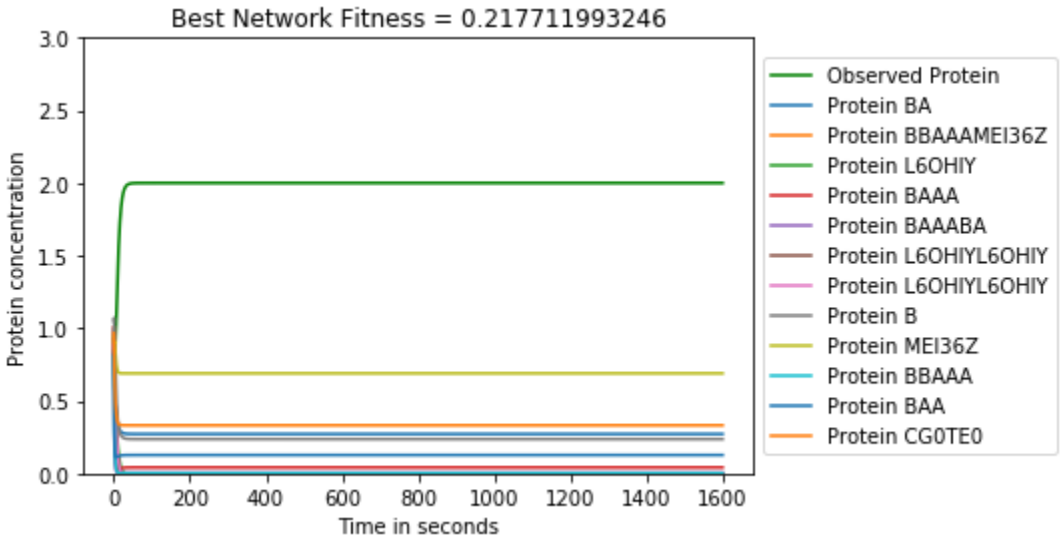
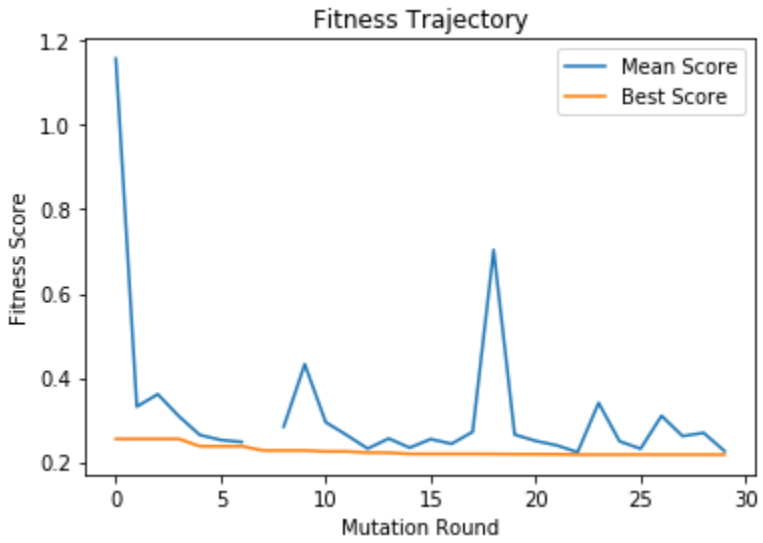


Figure 21. Changes in fitness over the course of mutation and selection in the system evolved with the modified oscillator fitness function.



4 Discussion

Although the authors mentioned that several oscillating networks were evolved that did not rely on transcriptional regulation, the lack of success experienced in this project suggests that at least some transcriptional regulation is necessary to along the path to evolve an oscillating network. One criticism that could be leveled against the attempt here is that the number of rounds of selection and mutation were insufficient. However, the procedure was repeated several times with a larger set of networks and up to 300 rounds of mutation. None of the networks evolved during this procedure showed even the beginnings of oscillatory behavior. The authors found that oscillating behavior usually appeared after the 260th generation (Francois and Hakim, Design of Genetic Networks 2004, 583). What this failure suggests is that mutual transcriptional regulation is a critical step in developing an oscillating network, and it is difficult to impossible to evolve a successful oscillator without it.

With this conclusion, we see again the value and utility of the evolution in-silico method for providing replicable, experimental validation of claims such as the necessity of transcriptional regulation in obtaining oscillating levels of proteins. The importance of transcriptional regulation hardly come as a surprise, but it can now be proven to necessary.

Future Directions:

Now that a working program to evolve networks through successive selection and mutations has been devised, it can be applied to new questions and areas of interest in biochemistry. Once a behavior can be identified and described through a fitness function, a network that exhibits it can be evolved using the programs developed in the course of this project. This makes possible experimental verification of claims that were previously relegated to speculation. As the literature reviewed at the beginning of this project shows, this method can be successfully applied to areas as diverse as embryonic development and the functioning of the immune system. A future application of this method might include attempting to model the mutations seen in cancer. Cancers are known to mutate in response to drug regimens, and this poses a significant challenge for treatment. Currently, the treatment is continued until symptoms show it to be ineffective, and it is not always clear what treatment should be tried next. While all of this goes on, the patient is enduring side effects for questionable benefit. A mutating network could be fit to known rates and types of mutations seen in cancer and then selected on by the traits known to confer resistance to a given treatment. This might allow for better prediction of when and how a cancer will become resistant to a given treatment regimen.

References

Francois, Paul. 2014. "Evolving Phenotypic Networks in Silico." *Seminars in Cell and Developmental Biology* (Elsevier) 35: 90-97. doi:10.1016/j.semcdb.2014.06.012.

Francois, Paul, and Eric D Siggia. 2008. "A Case Study of Evolutionary Computation of Biochemical Adaptation." *Physical Biology* 5 (5): 1-12. doi:10.1088/1478-3975/5/2/026009.

Francois, Paul, and Eric D. Siggia. 2012. "Phenotypic Models of Evolution and Development: Geometry as Destiny." *Current Opinion in Genetics and Development* (Elsevier) 22 (6): 627-633.

Francois, Paul, and Vincent Hakim. 2004. "Design of Genetic Networks with Specified Functions by Evolution in Silico." *Proceedings of the National Academy of Sciences of the United States of America* 101 (2): 580-585. doi:10.1073?pnas.0304532101.

Francois, Paul, Vincent Hakim, and Eric D. Siggia. 2007. "Deriving Structure From Evolution: Metazoan Segmentation." *Molecular Systems Biology* (EMBO and Nature Publishing Group) 3 (154): 1-9. doi:10.1038/msb4100192.

Lalanne, Jean-Benoit, and Paul Francois. 2013. "Principles of Adaptive Sorting Revealed by In Silico Evolution." *Physical Review Letters* 110 (21): 1-5. doi:10.1103/PhysRevLett.110.218102.

Tufcea, Daniel E., and Paul Francois. 2015. "Critical Timing without a Timer for Embryonic Development." *Biophysical Journal* (Biophysical Society) 109 (8): 1724-1734. doi:10.1016/j.bpj.2015.08.024.