EC500 D1 - Computational Synthetic Biology For Engineers

Homework 1
Spring 2017

February 7, 2017

Genetic Circuit Design

Due: 03/24/17

Overview and Background

One of the biggest challenges in Genetic logic synthesis is "Signal Matching". Unlike digital logic gates in electronics which have a system-wide 'High' or 'Low' value, genetic logic gates have unique 'Response Functions'. Response functions specify how the input signals of a genetic gate are mapped to its output signal. To learn more about this, please read Papers 1 and 2 in the **Suggested Reading Section**.

This homework will use Cello - A Genetic logic compiler, that can generate a genetic circuit design for a given Boolean function. The input for Cello, is a verilog file (Verilog is a hardware description language used to model and describe electronic systems), which describes Combinational Boolean Logic. Cello then uses data from a library called a UCF (User Constraint File), which contains empirically determined formulas for genetic gates. Each such genetic gate (often referred to as a repressor) can perform either a logical *NOR* operation or a logical *NOT* operation. Each gate can have either 1 or 2 inputs, but has only 1 output. If the repressor has 1 input, the output will be the logical negation of the input. If the repressor has 2 inputs, the output will be the logical disjunction negation of those two inputs. These repressors are used to synthesize a logic circuit that can realize the Boolean function specified in the verilog file.

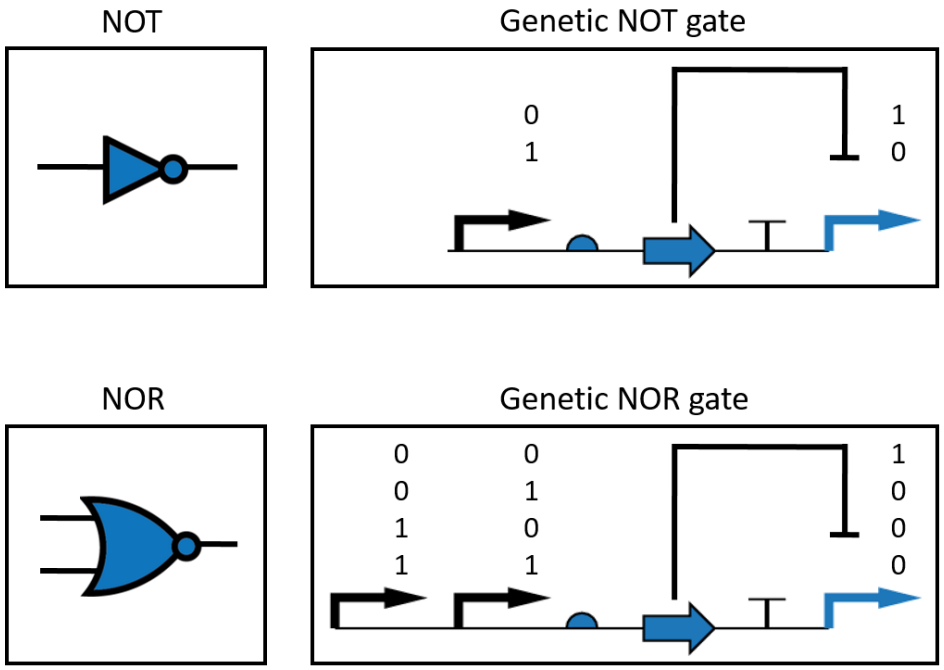


Figure 1: NOT and NOR gates used in Cello

Response Functions

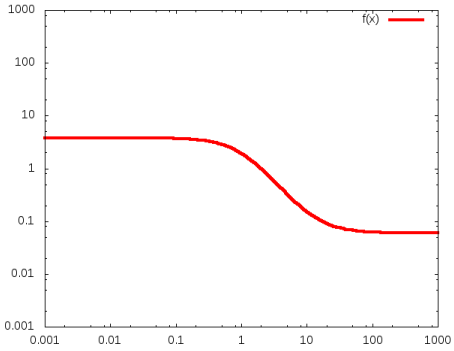
A response function:

$$\phi : \mathbb{R}^m_{\geq 0} \rightarrow \mathbb{R}_{\geq 0} \tag{1}$$

describes how the m Real value inputs, are mapped to a single real value output.
In Cello, a typical response function has the following template:

$$y = ymin + \frac{ymax - ymin}{1.0 + (x/K)^n} \tag{2}$$

The following figure illustrates one such form of the response function



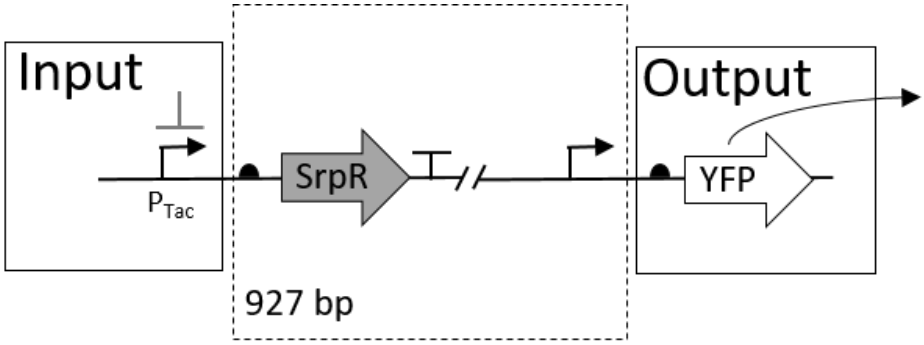
The x axis represents the input to the response function and the y axis, the output.(note that it has a sigmoidal (S-shape) form)

- ymax = max(y)
- ymin = min(y)
- n = slope of the curve
- K = K_d of the curve. (the distance between 0 to the mid-point of the max slope in the x axis)

Note: In Cello, if a repressor has 2 inputs, the value of the two inputs are added to get the value of x.

Example

Consider the following NOT gate:



In the absence of a small molecule (IPTG), the pTac can be "turned OFF" or will be in "LOW" state. Since the promoter is OFF, the output promoter is not repressed and we can see high protein expression. This is illustrated in the graph below shown in Fig 2

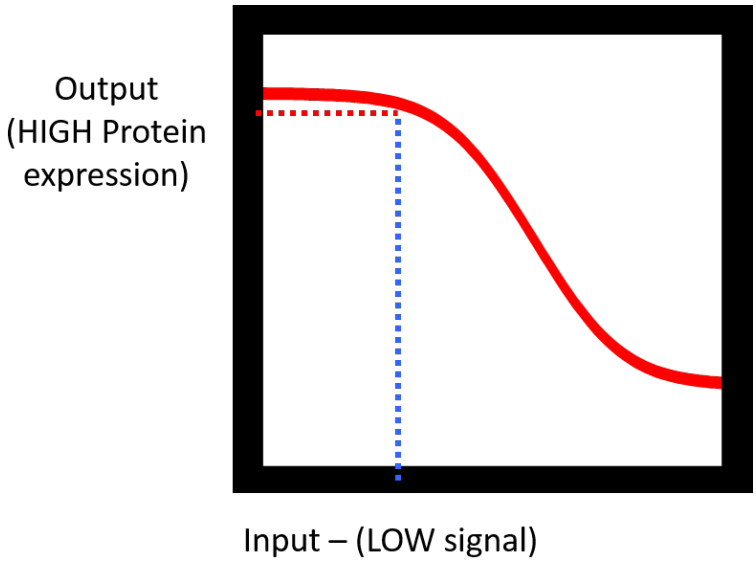
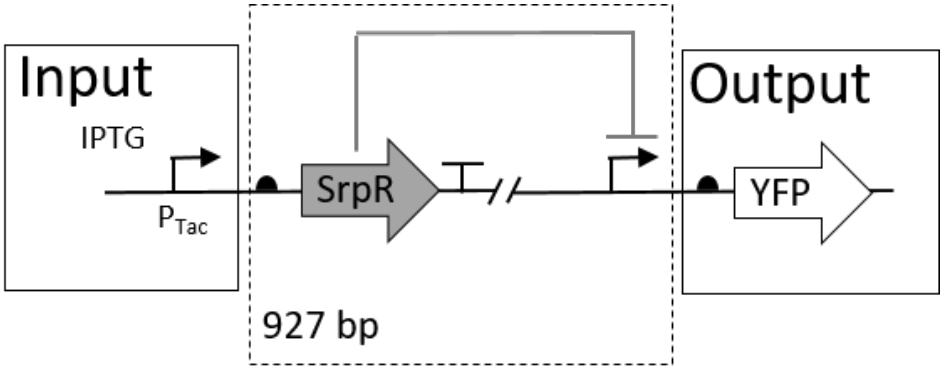


Figure 2: NOT Gate with LOW input

Similarly, in the presence of IPTG, pTac is now "ON".



The protein produced by SrpR represses the output promoter and low output protein expression is observed as show in Fig 3.

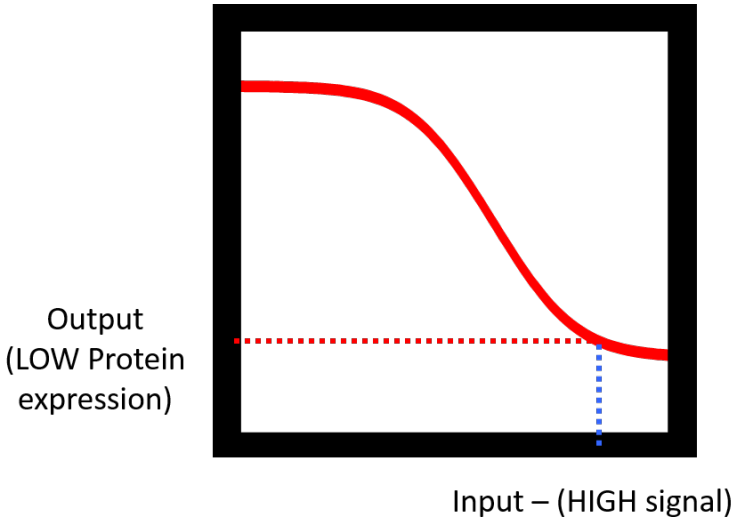


Figure 3: NOT Gate with HIGH input

Scoring a Gate

Since genetic logic signals do not have a well-defined high or low states, it is challenging to define how a genetic logic circuit implements a Boolean function. One way to counter this challenge is to compute how "close" a genetic gate's behavior is to a Boolean function. This closeness is defined by assigning a score to a gate. Scoring a gate is computed using a simple formula:

$$Score = \log\left(\frac{Lowest\ Predicted\ ON\ state}{Highest\ Predicted\ OFF\ state}\right) \tag{3}$$

(Note: This is log to the base 10)

Example 1:

Expected Boolean Function $y = a \oplus b$

Circuit 1:

a	b	y	value
0	0	0	0.01
0	1	1	15
1	0	1	16
1	1	0	0.02

$Score = \log\left(\frac{15}{0.02}\right) =$ **2.875061**

Circuit 2:

a	b	y	value
0	0	0	0.3
0	1	1	5
1	0	1	10
1	1	0	0.1

$Score = \log\left(\frac{5}{0.3}\right) =$ **1.22185**

Circuit 2 has a lower score as compared to Circuit 1.

Circuit 3:

a	b	y	value
0	0	0	0.3
0	1	1	0.4
1	0	1	10
1	1	0	2

$Score = \log\left(\frac{0.4}{2}\right) =$ **-0.698970**

This circuit has a negative score. Hence it does not realize the Boolean function at all.

Example 2:

You may have noticed, that a simpler abstraction (to describe if a genetic circuit can realize a Boolean function), is: $ON_{min} > OFF_{max}$. However, this definition allows a genetic circuit to realize multiple Boolean functions. Let $z = f(a, b)$ and have the values: {0.03, 0.01, 0.02, 15}.

a	b	$(a \bullet b)$	$(a \oplus b)'$	$(a + b')$	1	0	$Value$
0	0	0	1	1	1	0	0.03
0	1	0	0	0	1	0	0.01
1	0	0	0	1	1	0	0.02
1	1	1	1	1	1	0	15

Note: $+$ denotes logical disjunction (OR), $\bullet$ denotes logical conjunction (AND), $\oplus$ denotes Exclusive OR and $'$ denotes logical negation (NOT).
Since z could satisfy $(a \bullet b)$, $(a \oplus b)'$, $(a + b')$, 1 and 0, calculating the score helps find out which function it can realize the best.

Expected Boolean Function $z = (a \oplus b)'$
Circuit 1:

a	**b**	**z**	**value**
0	0	1	0.03
0	1	0	0.01
1	0	0	0.02
1	1	1	15

$Score = \log\left(\frac{0.03}{0.02}\right) =$ **0.176091**

Expected Boolean Function $z = (a \bullet b)$
Circuit 2:

a	**b**	**z**	**value**
0	0	0	0.03
0	1	0	0.01
1	0	0	0.02
1	1	1	15

$Score = \log\left(\frac{15}{0.3}\right) =$ **2.698970**

These values realize $(a \oplus b)'$ better than $(a \bullet b)$. To learn more about this, refer to Section III-C, "How a Genetic Circuit Implements a Boolean Function" from Paper 1 in the Suggested Reading section.

Enhancing a Repressor

As stated previously, each genetic gate has a unique property. Hence the response function for each repressor is different. This often leads to signal mismatching. Sometimes, when a "HIGH" signal is expected, the output is "LOW" (and vice-versa). A great repressor has high dynamic range (high ymax, low ymin) and a big slope. This can be achieved via protein engineering of repressors. Unfortunately, this is very difficult to do.
The response function can also be modified via DNA-engineering. This involves using different DNA-parts/components (such as using a different promoter or RBS).

Consider a simple repressor x with the following properties:

n=1.6, K=1, ymax=3.8, ymin=0.06

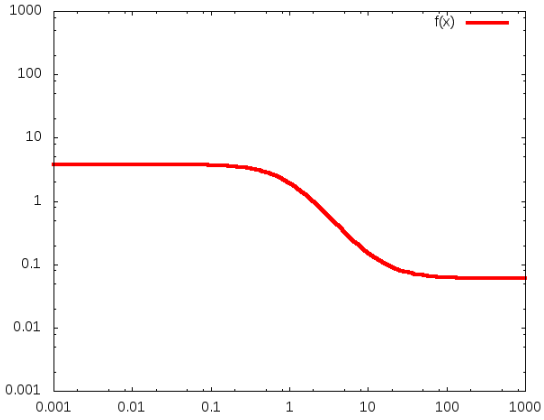


Figure 4: A typical Response Function

Note, that this is on a log scale. The the range of the x and y axis is 10^{-3} to 10^{3}.

The following section will look into a few DNA-engineering and Protein-engineering options

DNA-Engineering

- **Using a more active Promoter:**
  If the promoter is made more active, both the ymax and ymin can increase. (Since this is log scale, this increase will be a multiplicative increase. In this case the increase is 400%)

  n=1.6, K=1, ymax=15.2, ymin=0.24

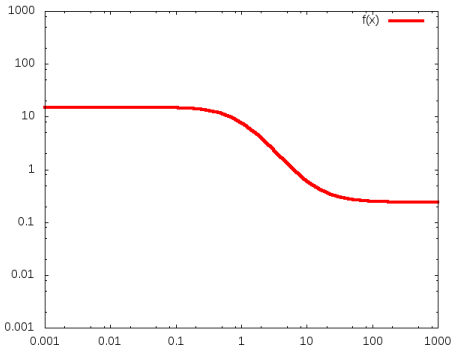


Figure 5: Promoter more active

- **Using a less active Promoter:**
  Similarly by making the promoter less active, both the ymax and ymin can decrease. (In this case, it is 25% of the original value)

  n=1.6, K=1, ymax=0.95, ymin=0.015

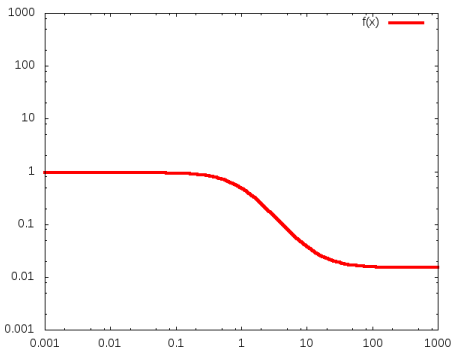


Figure 6: Promoter less active

(To make the distinction more obvious, the images have been provided separately)

- **Weaker RBS**
  By using a weaker RBS, the value of K can be increased.

n=1.6, K=15, ymax=3.8, ymin=0.06

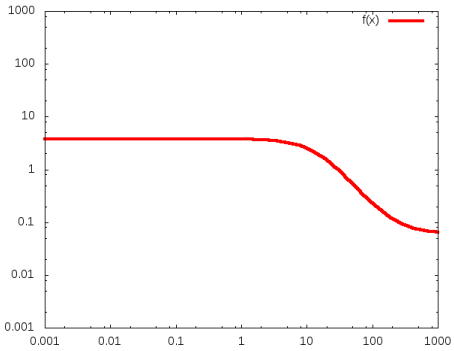


Figure 7: Weak RBS

- **Stronger RBS**
  And by using a stronger RBS, the value of K can be decreased.

n=1.6, K=0.005, ymax=3.8, ymin=0.06

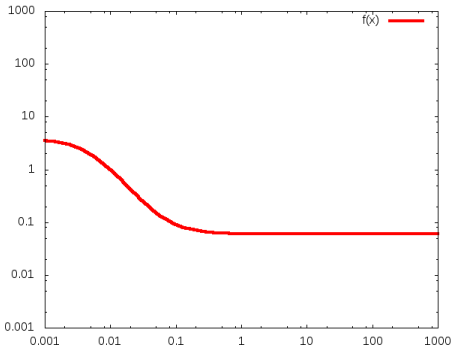


Figure 8: Strong RBS

Protein-Engineering

- **Stretch**
  By Stretching, one could increase ymax and decrease ymin. However, this is very difficult to do.

n=1.6, K=1, ymax=15.2, ymin=0.015

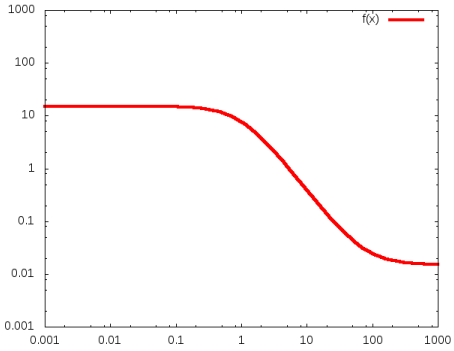


Figure 9: A repressor with high dynamic range

- **Increase the slope of the curve**
  Another difficult task is increasing the slope of the transfer curve.

n=10, K=1, ymax=15.2, ymin=0.015

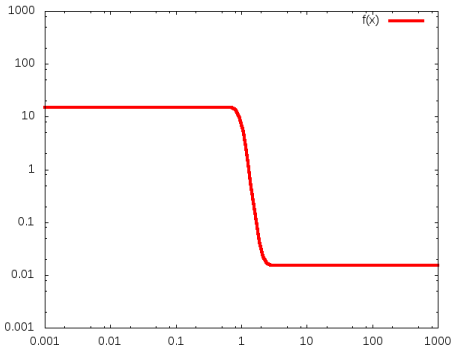


Figure 10: High slope

Goal

The score of a circuit is determined by how well the circuit can realize a logic function. This is directly related to the lowest value among the expected 'HIGH' signals w.r.t ymax of that repressor and the highest value among the expected 'LOW' signals w.r.t ymin of that repressor. To learn more about this, please read Section III, Part C, topic: 'How a Genetic Circuit Implements a Boolean Function' from Paper 1 in the **Suggested Reading Section**.

The goal of this homework is to come up with a software tool that can identify the best repressor candidate(s) for which such protein engineering techniques can be applied, and to find out how much of a gain in score is achieved by applying one or more of the techniques mentioned above.

Objectives

Given

1. **verilog.v** : Input Verilog file for Cello.
2. **Eco1C1G1T1.UCF.json** : A JSON representation of the Library used by Cello.

Using Cello

To try out cello, please go to **www.cellocad.org**
You can create a new user account and try out the features of the software tool.

UCF (User Constraint File)

Cello comes with a custom UCF (Eco1C1G1T1.UCF.json). To view the UCF file, go to 'Options'. To view the response functions, select 'response_functions' under 'View Collections'. You can also download the UCF file and make changes and upload a custom UCF file.
If you change the parameters of a response function, the on_threshold and off_threshold values must also be adjusted accordingly:

$$x = k * \sqrt[n]{\frac{ymax - y}{y - ymin}} \tag{4}$$

(Note: It is the n^{th} root)

$$off\_threshold = x \mid y = ymax/2 \tag{5}$$

$$on\_threshold = x \mid y = ymin * 2 \tag{6}$$

Cello Outputs

After you run Cello, you can download all the results. All files are named based on the name of the design specified by the user. The following files will be used by your software tool:

- DesignName_A000_bionetlist.txt
  Specifies the assignment of the logic circuit.
- DesignName_A000_wiring_agrn.png
  A diagram of the abstract genetic regulatory network
- DesignName_A000_logic_circuit.txt
  Specifies scores for each gate as well as the score for the entire circuit.
- DesignName_inputs.txt
  Specifies the input promoters used as well the ON and OFF values of the promoters. (These are the inputs to your circuit)
- DesignName_outputs.txt
  Specifies the output promoter used.

There are two important points to consider here:

1. The Score specified in Cello is not in log base 10. Instead it is simply ON_{min}/OFF_{max}.
2. The scores have been rounded off. Hence the value specified may not be exact. Read the following section to see an example.

Example

Please check the example_inputs.txt, example_outputs.txt, example_wiring_agrn.prn, example_bionetlist.txt, and example_logic_circuit.txt files in the HW1 folder. This section will help you understand how to analyze a sample Cello output.

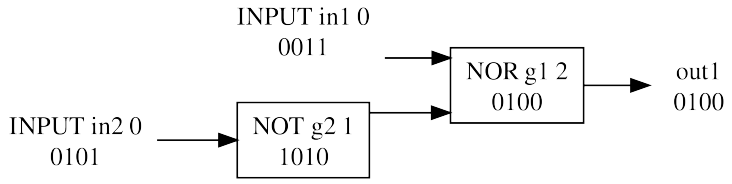


Figure 11: example_wiring_agrn.png : The wiring diagram of the Circuit

Inputs (from example_inputs.txt)
pTet, 0.0013, 4.4
pLuxStar, 0.025, 0.31

Output (from example_outputs.txt)
YFP

Assignment (from example_bionetlist.txt)

pLuxStar	0011	
pTet	0101	
P1_PhlF	pSrpR	pLuxStar
S1_SrpR	pTet	
YFP	pPhlF	

Response Functions (from UCF.json)

Gate	ymax	ymin	n	k
P1_PhlF	3.9	0.01	4	0.03
S1_SrpR	1.3	0.003	2.9	0.01

Equivalent Circuit diagram

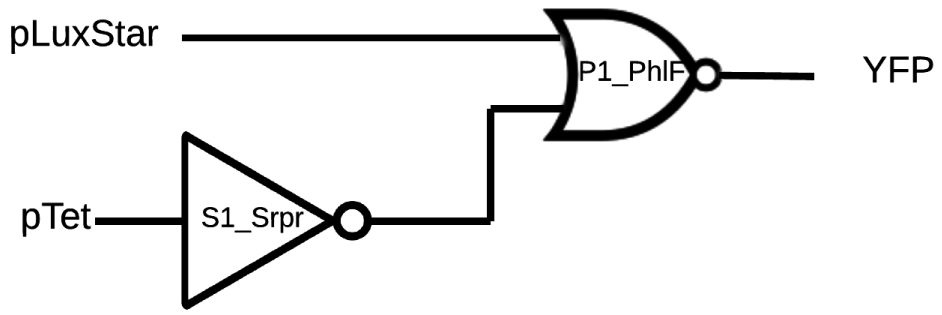


Figure 12: Circuit Diagram with assignments

Calculating the score

PLuxStar Bool	PLuxStar Val	pTet Bool	pTet Val	S1_SrpR	P1_PhlF
0	0.025	0	0.0013	1.296515	0.010001
0	0.025	1	4.4	0.003000	2.221688
1	0.31	0	0.0013	1.296515	0.010000
1	0.31	1	4.4	0.003000	0.010328

Understanding the Table:
Consider the first row of the table. The response function for S1_SrpR is given by:

$$y = 0.003 + \frac{1.3 - 0.003}{1 + (\frac{x}{0.01})^{2.9}} \tag{7}$$

The the value of x, is the value of pTet, which is 0.0013. The value of y = 1.296515. Similarly the response function for P1_PhlF is

$$y = 0.01 + \frac{3.9 - 0.01}{1 + (\frac{x}{0.03})^{4}} \tag{8}$$

Since the inputs of P1_PhlF is pLuxStar and the output of S1_SrpR, the value of x is the sum of those two values, which is 0.025 + 1.296515 = 1.321515. When this value of x is used in equation 8, the value of y (the output of P1_PhlF) is 0.010001.

Understanding the score of the Circuit:
Consider the column "P1_PhlF". Since P1_PhlF is connected to the output gate YFP, the score of both the gates are the same. Hence the score of the circuit is $= \log(\frac{2.221688}{0.010328}) =$ **2.332667** and $\frac{ON_{min}}{OFF_{max}} = 215.1130906$.

Cello API

Please refer to "CelloAPI.pdf" to learn how you can connect to Cello.

Your Software Tool

Objective

You have to create a tool that can connect to Cello either using the Cello API, or Curl commands or any custom API, and run Cello for a given design and UCF. The tool must then identify the best/ideal repressor(s) for which protein or DNA engineering can be applied. After modifying the parameters for the response function, the tool must then calculate the improved score (which can be verified by re-running Cello with the modified UCF file).

Inputs

- Verilog File
- UCF File - This will be the UCF used by Cello.
- n - The number of repressors for which protein or DNA engineering can be applied. This should be less than or equal to the number of repressors in the UCF file.

Outputs

- The modified score (along with the percentage gain in score)
- The list of repressors that have been modified and the operations applied on them.
- The modified UCF file.

Operations and Rules

You can apply the following operations on a repressor to modify its response function.

- Stretch (x):

  $ymax_{new} = ymax * x$

  $ymin_{new} = \frac{ymin}{x}$

  x can be at-most 1.5

- Increase Slope (x):

  $n_{new} = n * x$

  x can be at-most 1.05

- Decrease Slope (x):

  $n_{new} = \frac{n}{x}$

  x can be at-most 1.05

- Stronger Promoter (x):

  $ymax_{new} = ymax * x$

  $ymin_{new} = ymin * x$

- Weaker Promoter (x):

  $ymax_{new} = \frac{ymax}{x}$

  $ymin_{new} = \frac{ymin}{x}$

- Strong RBS (x):

  $K_{new} = \frac{K}{x}$

- Weaker RBS (x):

  $K_{new} = K * x$

- Swap gates
  You can also swap the repressors assigned to calculate a new score for the circuit.

- Swap Inputs
  You can also swap the input promoters (or the order of the input promoters to see if you get a different/better score). Note, the OFF and ON values of an input promoter cannot be changed.

Scoring

Note, your tool will be tested with a custom UCF file. Total Points for the Homework: 120 points.

- Turn in source code to Blackboard: 10 points
- Provide documentation (with a clear example of how your tool must be installed/compiled/used): 20 points
- Well organized code, with comments: 10 points
- Efficiency of your tool: 60 points
  This will be calculated based on various factors:
  - Time taken for your code to run
  - The number of operations used to enhance the score and the actual gain in score.
    * The fewer the number of operations used, the better your score will be. However, you can use Swap gates and Swap inputs as many times as you'd like.
    * You will score more points for using DNA-engineering operations like using Stronger Promoter, Weaker Promoter, Weaker RBS, Stronger RBS instead of protein-engineering operations like Stretch, Increase Slope, Decrease Slope

- Report (minimum 1.5 pages, maximum 6 pages): 20 points
  This report should explain your algorithm, the complexity of your algorithm, specifics of your tool (programming language used, platform, APIs, etc) and the challenges faced.

Bonus Section

NOTE: This homework has a lot of bonus points. These points can significantly boost your score.

- **2 Points**: Find and report a bug in Cello.
  Please follow the bug reporting guidelines for this.
- **2 Points**: Find and report a bug in the Cello API.
  Please follow the bug reporting guidelines for this.
- **30 Points**: Created a GUI or a Web application. (This will be graded on a scale of 1-30)
  Please note: slapping a few UI elements together does not constitute a webapp or GUI. The UI must be easy to use and must 'look decent'.
- **5 Points**: Created a script that can easily load and modify ucf.json
- **20 Points**: Created a GUI or a Web interface that can easily load and modify ucf.json (same UI concept applies here)
- **20 Points**: Created a custom Python Cello API or a Cello API in any other programming language.
  (If you create a Python API, please state clearly how your API is an improvement on the current API)

Reporting a Bug

Typical Bug reports include 4 key aspects:

- Description
- Steps to recreate the bug
- Expected Result
- Observed Result

Example:

- Description: The purpose of this issue is to show how a standard issue/bug report should look like.
- Steps to recreate the bug: Detail the steps to recreate the bug. Eg:
  1. Go to the "Options Page"
  2. Select 'response_functions' in the "View Collection" option
- Expected Result: Response functions details should load.
- Observed Result: Page goes blank.

Please email your bugs to: prash@bu.edu

Submission

- All source code must be turned in to Blackboard in a zip file.
- **Please make sure you include a file README.txt in the root of your zip file.** This file should contain clear instructions for building, compiling and using your code. The file should also include all the software tools/platforms (along with the version) required for your code to run.
- DO NOT UPLOAD BUILD OR NATIVE OS CONFIGURATION FILES OR IDE SPECIFIC FILES. You must only include the source code and other **required** files needed to compile and build your code.

- Please make sure you include your report in the root folder of your project. Please name the file: "Report.pdf" (It should be a pdf file).
  - Please make sure you include your name (your real name... e.g: Bruce Wayne) in the report. (The name of both team members should be included in the report)

Suggested Reading

1. Vaidyanathan, P., Der, B. S., Bhatia, S., Roehner, N., Silva, R., Voigt, C. A., & Densmore, D. (2015). A Framework for Genetic Logic Synthesis. Proceedings of the IEEE, 103(11), 2196-2207.
(This Paper should help understand the basic framework, terms and definitions of Logic Synthesis applied for Genetic circuits. Specifically, Sections 1,2,3 as well as Fig 2 will be helpful)

2. Yaman, F., Bhatia, S., Adler, A., Densmore, D., & Beal, J. (2012). Automated selection of synthetic biology parts for genetic regulatory networks. ACS synthetic biology, 1(8), 332-344.
(The Results section as well as Fig 5 will be useful)

3. Stanton, B. C., Nielsen, A. A., Tamsir, A., Clancy, K., Peterson, T., & Voigt, C. A. (2014). Genomic mining of prokaryotic repressors for orthogonal logic gates. Nature chemical biology, 10(2), 99-105.