Final Project Report

Brian Mahabir, Landon Kushimi, Brian Homoon Jung, Zian Wang

Our project provides a streamlined process for giving users suggestions to use other available terminator promoter pairs that is the most cost effective by looking at the design and constraints from Eugene and using Double Dutch. We have designed a Graphical User Interface that interacts with Eugene and Double Dutch to give the user a quick way to determine if their current structural specification can be improved by using a different terminator or promoter. It should also be able to do that for ribosome binding sites as well as other coding sequences.

The code is written in Python 3.7.9 and uses many preinstalled libraries to get the job done. The first major library the code uses is tkinter which is the main library to create the GUI. Scipy is used for data analysis if necessary, for the mathematics behind Eugene and Double Dutch. Finally, we have selenium which is arguably the most important library that our code uses. Selenium interfaces with front end graphical user interfaces to operate the code from the front end and also is able to run both Double Dutch and Eugene in the backend.

To preface why we decided to use selenium instead of a different method, we have to delve a little bit into our trouble shooting problems. Originally, we were going to design the code to interact with Eugene and Double Dutch from the backend to both enter the user's inputs and extract the outputs. However, Eugene ended up being too cumbersome to sift through. We opted to go for miniEugene instead. However, miniEugene could not be run locally due to CORS errors surrounding jQuery and google analytics that we spent too much time trying to figure out. Double Dutch could not be run through a webserver since its server was down. We had to

compromise with using selenium in tandem with python requests to get both Eugene and Double Dutch running the way we wanted.

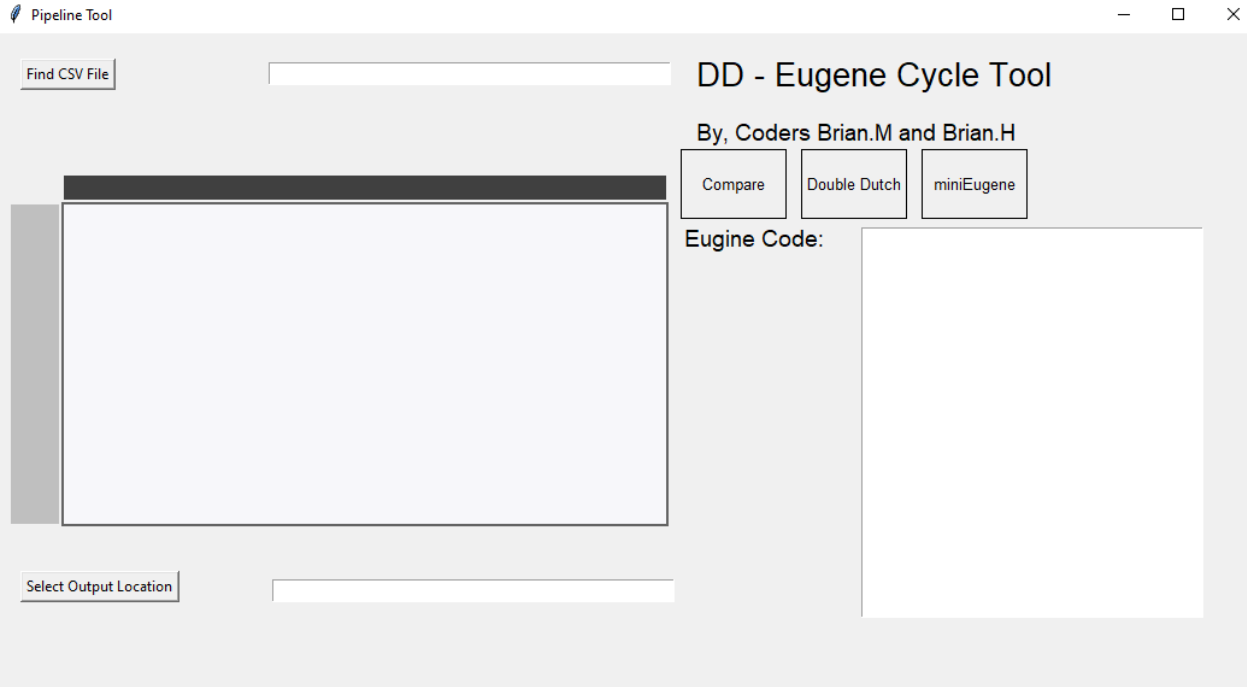

figure 1. Example of the pipeline tool

First the user can code Eugene code in our built-in text editor. Code mirroring is not supported on python and highlighting each technical word would be too hard so the text editor is in plain text. However, the Eugene code is able to be parsed later through our written function to extract the coding sequences and other factors (starts with c or is linked to a coding sequence via interaction constraint and pairing constraint). This will be later used to be added to the csv file as the Double Dutch input.

The user can then hit miniEugene button and input their constraints to get an output. These outputs will show them a graphical representation of designs that match the structural specification they gave.

The user will then hit the Double Dutch button which will run the Double Dutch code locally on their machine. The user can then enter in the same credentials to get a plethora of different models. The code will also take save this as a csv file and give the library as well. The user can then pick a different terminator and promoter pair to use instead of the current one.

While not completely finished, the solve button is being designed to prune the inconsistencies between the outputs of Eugene and Double Dutch to get rid of Eugene design that suggest incompatible pairings as well as structures. We want it to be an automatic process, and more of a “cycle” where an user can repeat this process to reach the most cost effective design that also matches their structural specification.