Rishab Jain
Summer Intern @ Lattice Automation
July 2021

Introduction:

Optimizing synthetic DNA sequences for maximum expression in a cell factory is an important segment of heterologous expression. However, current codon optimization techniques may involve overwriting codon frequencies based on that of the host. The ICOR approach proposes recurrent-neural-networks to learn the codon usage bias present in a subset of 7,000 highly-expressed *E. coli* genes. 40 benchmark sequences were selected from papers related to heterologous expression in *E. coli* and codon optimization. In this report, a comparison will be shown for the benchmark sequences optimized by the ICOR model, sequences optimized by the naive and super naive approaches, and the original sequences themselves.

Algorithm Overview:

Super Naive Approach

The super naive approach -- code here: GitHub -- utilizes a look-up table that defines amino acids and their corresponding codons.

Sample:
"A": ["GCT GCC GCA GCG"],
"R": ["CGT CGC CGA CGG AGA AGG"],
"N": ["AAT AAC"],

For each amino acid present in the benchmark sequence, the super naive algorithm will randomly select a corresponding codon.

Naive Approach

The naive approach -- code here: GitHub -- utilizes a look-up table that defines amino acids and their corresponding codons, **along** with probabilities for how frequently these codons appear.

Sample:
"A": (["GCG", "GCA", "GCT", "GCC"],[0.34, 0.22, 0.17, 0.27]),
"R": (["AGG", "AGA", "CGG","CGA","CGT","CGC"],[0.03, 0.05, 0.1, 0.07, 0.37, 0.38]),
"N": (["AAT", "AAC"],[0.46, 0.54]),

For each amino acid present in the benchmark sequence, the naive approach uses the np.random.choice method to select a corresponding codon given the probabilities that they occur in *E. coli*.

Such probabilities were normalized to sum to exactly 1.0.

Rishab Jain
Summer Intern @ Lattice Automation
July 2021

ICOR Approach

The ICOR approach utilizes an RNN that contains BiLSTM layers. The model is trained on high-frequency *E. coli* genomes themselves. As a result, the model is able to learn the codon usage distributions of high-frequency sequences, while still keeping knowledge of subsequences and patterns for how the codons are used.

For each of the 40 sequences, inference was performed using the ICOR model in MATLAB. The output sequences were saved in the FASTA format.

Optimization Time:

Testing System:

- Intel i7-10700
- 32gb 2933 mhz RAM
- 3k/3k NVMe
- RTX 2060

Test was run on the 40 benchmark sequences.
Average length of benchmark sequences: 1687.65 nt
Average codons of benchmark sequences: 562.55 codons

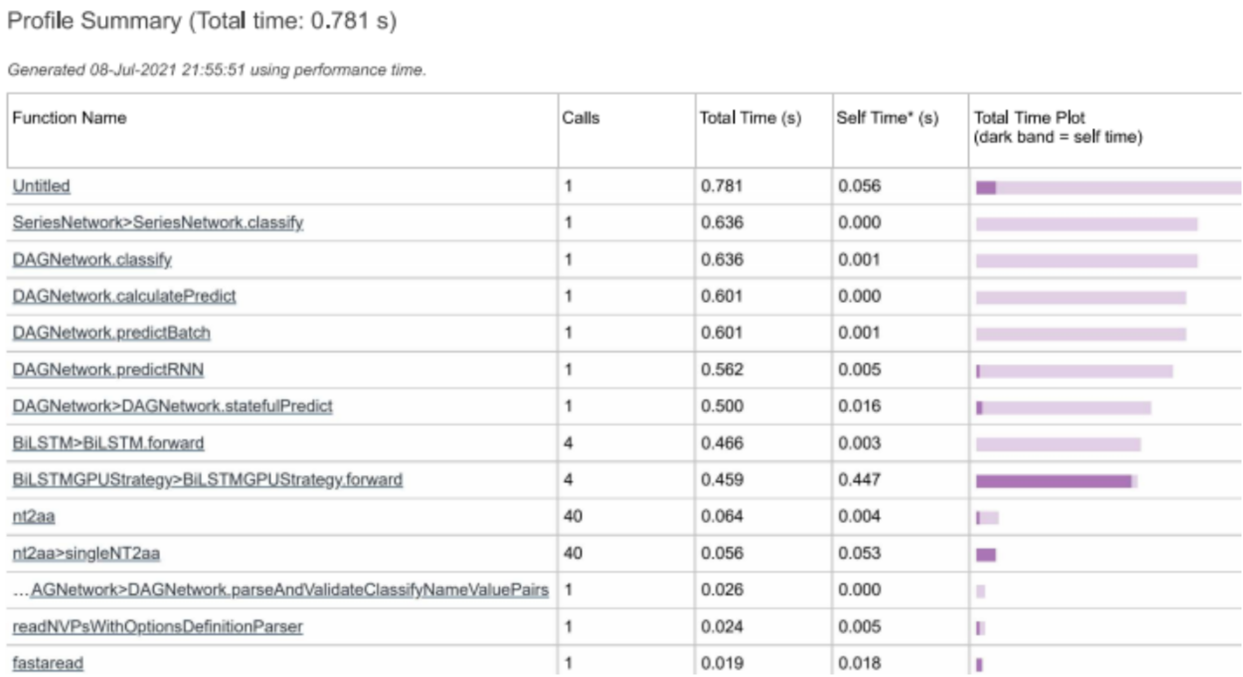
	Average Time (n. seq=40) (n. trials=3)	Time Per Seq.	Time Per Codon	Time per NT
Brute Force	**275587%** (749597ms) (750.517s + 749.821s + 748.455s) / 3	**18739.9ms** 749597ms/40	**1332.50ms** 749597ms/562.55	**444.166ms** 749597ms/1687.65
ICOR	**288%** (782ms) (0.772s + 0.807s + 0.766s) / 3	**19.5ms** 782ms/40	**1.39ms** 782ms/562.55	**0.463ms** 782ms/1687.65
Naive	**283%** (771ms) (0.761s + 0.764s + 0.787s) / 3	**19.3ms** 771ms/40	**1.37ms** 771ms/562.55	**0.456ms** 771ms/1687.65
Super Naive	**100%** (272ms) (0.271s + 0.274s + 0.272s) / 3	**6.8ms** 272ms/40	**0.48ms** 272ms/562.55	**0.161ms** 272ms/1687.65

Table 1. As it can be seen, the ICOR approach is very fast and finishes in only 0.782s for **40 sequences**. The inference time is extremely low. In an actual implementation for a customer, if just one sequence is being optimized, the output should be displayed in a matter of seconds. The main time for processing will be to load the model (as the actual inference is very fast).

In this testing, the model was already loaded into the workspace which is held in memory to some extent. In an end-to-end system, the model may be hosted on a server, or on a local file. In that case, it may take a few extra seconds.

However, from earlier testing with the ICOR GUI (developed in MATLAB), it is estimated that this would take 2-3 seconds to run.

ICOR Profile Summary from 1 run:



Profile Summary (Total time: 0.781 s)

Generated 08-Jul-2021 21:55:51 using performance time.

Function Name	Calls	Total Time (s)	Self Time* (s)	Total Time Plot (dark band = self time)
Untitled	1	0.781	0.056	
SeriesNetwork>SeriesNetwork.classify	1	0.636	0.000	
DAGNetwork.classify	1	0.636	0.001	
DAGNetwork.calculatePredict	1	0.601	0.000	
DAGNetwork.predictBatch	1	0.601	0.001	
DAGNetwork.predictRNN	1	0.562	0.005	
DAGNetwork>DAGNetwork.statefulPredict	1	0.500	0.016	
BiLSTM>BiLSTM.forward	4	0.466	0.003	
BiLSTMGPUStrategy>BiLSTMGPUStrategy.forward	4	0.459	0.447	
nt2aa	40	0.064	0.004	
nt2aa>singleNT2aa	40	0.056	0.053	
...AGNetwork>DAGNetwork.parseAndValidateClassifyNameValuePairs	1	0.026	0.000	
readNVPsWithOptionsDefinitionParser	1	0.024	0.005	
fastaread	1	0.019	0.018	

Figure 1. Profile summary from one run of ICOR inference. References calls and times for each of the functions. “Untitled” is the time taken to run the entire program.

Results:

Result benchmarks were calculated using GenScript’s Rare Codon Analysis Tool which can be accessed here: GenScript Rare Codon Analysis Tool.

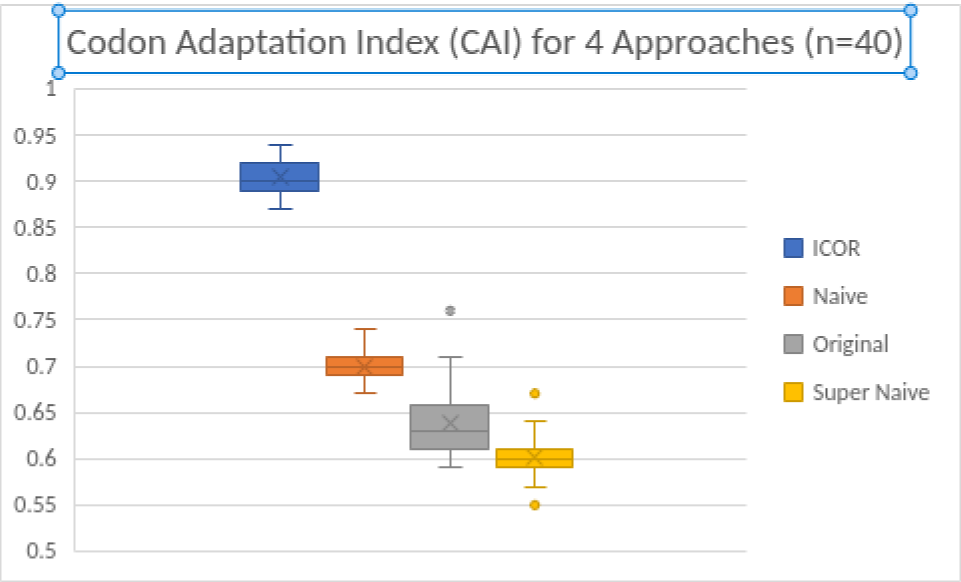


Figure 2. A set of 40 proteins were randomly selected from published works on protein expression in E. Coli in order to benchmark the performance of ICOR as compared to industry tools. These proteins came from a variety of origin organisms and had a mean codon adaptation index (CAI) of 0.638 with a standard deviation of 0.0386. ICOR optimization resulted in a mean CAI of 0.904 with an st. dev. of about 0.016, signifying a ~41.692% increase. This was statistically significant (p<0.0001) using a two-sample t-test. The super naive approach had a mean CAI of 0.602 and st. dev. of about 0.022. ICOR offered a ~50.21% increase compared to this approach. This was statistically significant (p<0.0001) using a two-sample t-test. Finally, the naive approach offered a mean CAI of 0.699 and an st. dev. of 0.0158. ICOR offered a ~29.32% increase in CAI compared to the naive approach. This was statistically significant (p<0.0001) using a two-sample t-test.

According to Genscript, the ideal value of CAI should be from 0.8-1.0. ICOR had a CAI in this range for 40/40 sequences. The naive approach albeit close had 0/40 genes in this range. The super naive and original genes had 0/40 genes in the range as well.

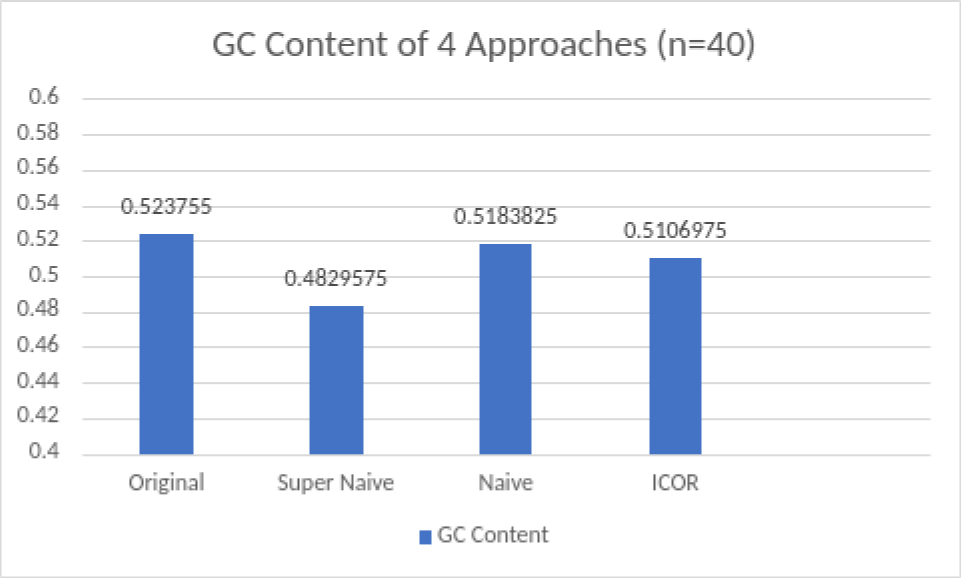


Figure 3. Sequence GC content between 40 to 60% is known to promote binding, and thus, maintain good expression. The ideal percentage range of GC content is between 30% and 70% according to GenScript. They state that “Any peaks outside of this range will adversely affect transcriptional and translational efficiency.” The ICOR approach’s mean GC content was approximately 51.07%. For the 40 benchmark sequences, the ICOR, Naive, and did not have **any** sequences outside of this range.

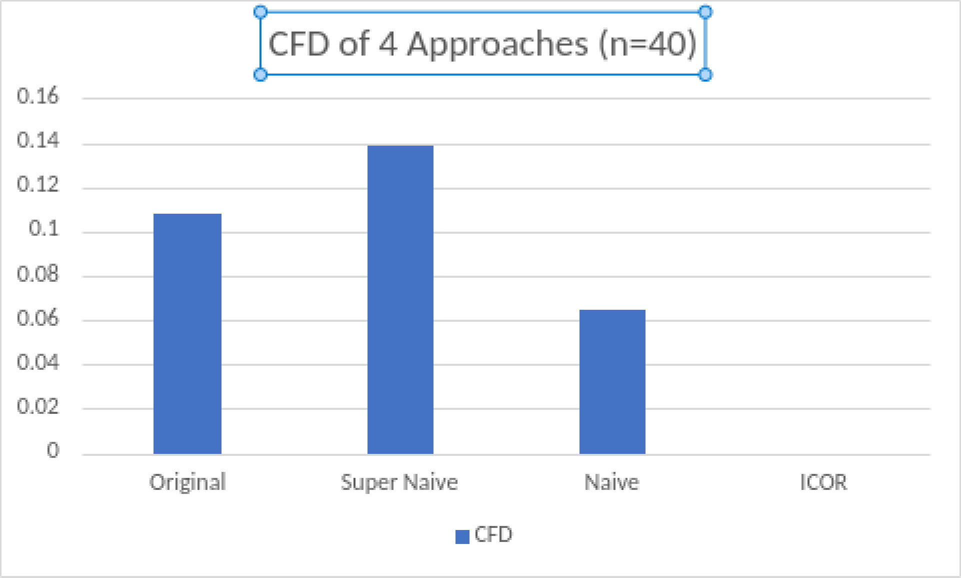


Figure 4. CFD (%) is a metric calculating tandem rare codons that can “reduce the efficiency of translation or even disengage the translational machinery.”[1] The ideal value according to GenScript is less than 30%. ICOR eliminates all tandem rare codons that could reduce

[1] https://www.genscript.com/tools/rare-codon-analysis

efficiency. When compared to three of the other approaches, ICOR's improvement was statistically significant (p<0.0001) using a two-sample t-test.

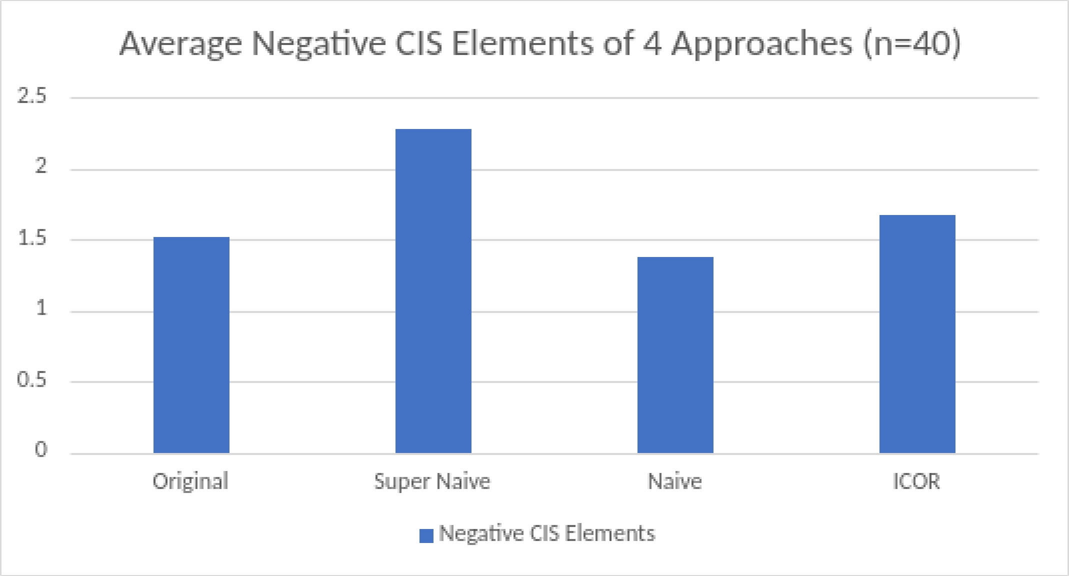


Figure 5. The ICOR, Naive, Super Naive, and Original approaches had 1.675, 1.375. 2.275, and 1.525 negative CIS elements respectively on average. For 25 of the genes, ICOR had "less than or equal to" negative CIS elements than the naive approach. For 24 of the genes, ICOR had "less than or equal to" negative CIS elements than the original approach. For 31 of the genes, ICOR had "less than or equal to" negative CIS elements than the super naive approach. For most of the genes, ICOR kept "less than or equal to" negative CIS elements than the other approaches. But then why does ICOR have a higher average? This is because three genes in particular: UBTF, PDCD11, and PAK1 each had 7 negative CIS elements which was quite a bit compared to the other sequences.

However, a different statistical test may be better for this metric. This is because the minimum number of negative CIS elements is 0 which means it not a normal distribution, and is a non-parametric distribution.

For most genes, ICOR's approach is superior. According to the 68–95–99.7 rule, the ICOR approach will only have these outliers for less than 5% of the genes. However, this doesn't seem to be a problem. Studies show that CAI is highly correlated with real-world protein expression, and because of the significant increase in CAI, the overall gene expression is still greater.

It’s possible that some of these negative CIS elements are also important for the cell metabolism. Negative CIS elements are known to negatively regulate gene expression, however, this may be important for keeping the cell from expressing even beyond the realm of “overexpression” that is typical for cell factories used in heterologous expression such as *E. coli.* Further, as seen in studies conducted for other codon optimization tools, although these negative elements are actually higher, the overall gene expression still remains increased.

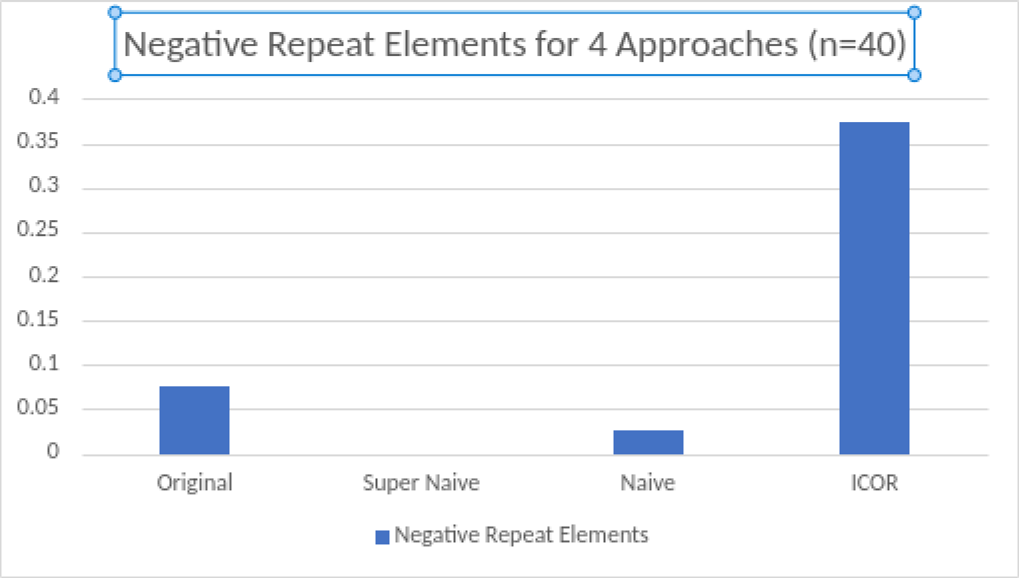


Figure 6. The ICOR, Naive, Super Naive, and Original approaches had 0.375, 0.025. 0, and 0.075 negative repeat elements respectively on average. For 34 of the genes, ICOR had a “less than or equal to” amount of negative repeat elements compared to the naive approach. For 33 of the genes, ICOR had a “less than or equal to” amount of negative repeat elements compared to the original approach. For 33 of the genes, ICOR had a “less than or equal to” amount of negative repeat elements compared to the super naive approach. However, a different statistical test may be better for this metric. This is because the minimum number of negative CIS elements is 0 which means it not a normal distribution, and is a non-parametric distribution.

The super naive approach is randomly selecting codons yet it has even less negative repeat elements than the original. This goes to show that this metric does not necessarily equate to more or less expression. ICOR is trained on highly-expressed genes of *E. coli* whereas the Original sequences are sequences from organisms that *are not E. coli*. This suggests that some highly expressed sequences will have negative repeat elements that may be direct or inverted repeats, but still contribute to the gene as a whole. Another interesting discussion is that there were two genes in particular that were massive outliers for the ICOR tool and strongly increased the number of repeat elements.