

Symbolic Regression for Translation Efficiency

Demo Marco, Di Benedetto Davide, Nadif Nizar

Abstract—Translation efficiency, or the amount of protein expressed in the cell from a predefined number of mRNAs, is a crucial biological measure in biotechnology, especially considering its application in mRNA vaccines, such as the one for COVID-19. Aiming to create a model that could serve as a fitness function in a competitive evolution contest against other models, we chose Genetic Programming not only for its efficiency and numerous successful applications but also for the possibility of biologically interpreting the resulting regression. After a series of optimizations, including feature and target linearization and Quantile-Based Binning, we obtained a model with $R^2 \approx 0.70$, which, given the scarcity of in vitro data and the biological complexity of the problem, we decided to consider it a proper result, usable as a tool. We also find a coherence between the position of attribute in the obtained formulas and the biological literature. The codebase can be found on GitHub (NizarNadif/symbolic-regression-rna).

NOWADAYS, in order to verify the efficiency of an mRNA vaccine in terms of protein production, it's necessary to perform *in-vitro* analysis of the codon sequences. This process is tedious and, as the datasets that will be presented in the following sections testify, is performed in few places. The initial idea of this project was to find a loss function for mRNA vaccines, but it soon became clear that the data necessary was insufficient: the first and only mRNA vaccines made are the COVID-19 ones, that produce the Spike Protein. So, the data that would be processed and used for whichever model chosen, would only be relative to a single protein; hence, it would surely overfit given any possible metric.

So, the focus soon shifted to a more general topic, such as the translation efficiency of mRNA strings to proteins. The problem tackled is codon degeneracy: going from RNA to a protein is straightforward, but not the inverse. An RNA string is composed by codons, that are triplets of nucleotides¹; given that, there are 64 possible codons, but each of them synthesizes one of the 22 α -amino acids, the building blocks of proteins². This mismatch is called codon degeneracy, and it implies that a protein can be expressed in multiple ways and, each one of those produces a different amount of proteins in the system that hosts it.

Once the focus shifted to codon degeneracy in the human body, the objective was identified into finding a symbolic regression for translation efficiency, that could be used to evaluate the "goodness" of a mRNA string in terms of protein production. The technique chosen is genetic programming, because of its versatility given a well-defined terminal set.

I. DATASET

A. Data Retrieval

To train GP, we decided to use several datasets; initially, for the first runs, we focused on individual, separate datasets to obtain meaningful results and to understand whether the problem was actually attributable to optimization via symbolic regression. The biggest obstacle was the lack of in vitro experiments that directly calculated or addressed Translation Efficiency as a metric usable by our model. Ideally, we would have liked to use only data from Human Related experiments, but unfortunately, this proved to be impossible due to the lack of such experiments. We will discuss the attribute synthesis phase in a later section of this paper, but it is important to clarify that a model like genetic programming obviously cannot take entire nucleotide sequences (on the order of tens of thousands of nucleotides) as input; instead, parameters that can be represented numerically must be calculated. Another challenge was normalizing the data afterwards: when we decided to merge the datasets to serve as input to the model, a simple normalization created problematic outliers that interfered with the rest of the distribution.

1) *E. Coli Dataset (Sanofi / mRFP)*: This was the first dataset we decided to consider, as it was the dataset used in the Sanofi paper as a benchmark for the CodonBERT model [1]. This dataset was obtained through a series of in vitro experiments to measure the amount of fluorescence due to the mRFP protein [2]; it was therefore obtained using a series of synonymous codons, meaning they code for the same protein but with a different sequence. This seemed like an excellent starting point for our model, not only because the measure of protein quantity obtained from the mRNA codon is very direct (fluorescence given by the amount of protein expressed), but also because the applications for which the model was designed would seek to solve similar problems: consider mRNA vaccines. The final processed dataset includes a total of 1,459 unique mRFP coding sequence variants, representing the full spectrum of synonymous optimization analyzed in the original study.

2) *Human 5' UTR Dataset (Sample et al.)*: The final dataset we decided to use refers to the work of Sample et al. (2019) [3] and consists of a vast library of synthetic human 5' UTRs, i.e., those regions of the string found before the CDS (coding sequence) that, although they do not affect the protein itself, influence its ability to bind to ribosomes and initiate translation. The data were obtained through a large-scale Massively Parallel Reporter Assay (MPRA) ($\sim 180,000$ sequences) using the Polysome Profiling technique to measure the Mean Ribosome Load (MRL), i.e., the amount of free

¹A, G and C for both DNA and RNA, then T and U respectively

²20 of those are actually degenerated, whilst 2 of them indicate the end of a protein and are univocally indicated by stop codons

ribosomes for each sequence. All these sequences were then complemented with the CDS reported in the paper, i.e., EGFP. This dataset allows the model to learn on metrics that do not apply directly to the CDS, such as Kozak strength.

3) *Yeast Dataset (Weinberg)*: The two datasets previously discussed had major limitations: either the Coding Sequence (CDS) was fixed, or the 5' UTR was completely absent; thus, a dataset differentiated by both these characteristics was needed. Although we distanced ourselves from studies related to the human species, we agreed on the need to use Weinberg's Yeast dataset [4] (also analyzed by Wint et al. [5]) containing data from in vitro experiments conducted on a large portion of the *S. cerevisiae* genome. This dataset was obtained through experiments aimed at improving the Ribosome Profiling technique; a close link was subsequently found between Translation Efficiency (i.e., our target) and the presence of optimal codons (CAI) and GC Content in the study by Wint et al. (2022) [5], which then guided our choice of terminal set attributes to use. Following the application of quality filters on basal expression ($RPKM > 5$), the final genomic dataset consists of 4,820 unique *S. cerevisiae* genes, offering representative coverage of the yeast transcriptome.

B. Pre Processing

1) *Individual Dataset Preparation*: Prior to entering the common pipeline, specific preprocessing steps were required to standardize the raw data formats (X) and target variables (Y) for each organism:

- **E. Coli Dataset (Sanofi / mRFP)**: This dataset, in particular, required no special preprocessing because we took it directly from Sanofi's GitHub (which used it to train CodonBERT [1]); it simply went through the normalization pipeline along with all the others.
- **Human 5' UTR Dataset (Sample et al.)**: In the case of this dataset, we decided to add the CDS of the target gene from the reference experiment (the EGFP protein) to the 5' UTR sequences so that we could calculate and use the CDS folding values along with the Kozak Strength. The target value we used was the Mean Ribosome Load (MRL) of the experiment in question [3]:

$$Seq_{total} = UTR_{5'} \oplus CDS_{EGFP} \quad (1)$$

- **Yeast Dataset Preparation (Weinberg/Wint)**: For the Yeast dataset, the data were slightly more difficult to re-construct because the experiment results were not directly related to Translation Efficiency (TE). We therefore had to retrieve the mRNA and ribosomal abundance data from [4] and then split them according to the formula:

$$TE = \frac{Ribo_RPKM + \epsilon}{RNA_RPKM + \epsilon} \quad (2)$$

To avoid outliers and filter out noise, we had to apply the condition $RNA_RPKM > 5$; This avoids both poorly expressed genes and divisor values close to zero. We then had to map the FASTA sequences to the IDs provided by the article.

C. Common Preprocessing Pipeline

In addition to the optimizations related to the individual datasets, it was necessary to build a pipeline to merge and normalize the data, in order to perform a complete run that covered all possible input facets to the model.

- **Quantile-Based Binning**: Since the datasets were all of different sizes and had target values that did not follow the same order of magnitude (e.g., Sanofi vs. Yeast), it became necessary to choose a minimum number of values to include in the training and test sets. We chose to use the same number of sequences for each dataset; this number was determined by taking the lower limit of size, namely the **1,459 sequences** of the Sanofi dataset [2]. The *Quantile-Based Binning* technique was used to extract the values to be considered. The datasets were sorted and discretized using quantiles (via Python); representative samples were then extracted from these bins to reach the target size, trying to maintain the mean value inside the bin, and avoid repeated values. This resulted in three datasets with sensible distributions and the same size.
- **Pre-Merge Partitioning**: Obtaining datasets of consistent size allowed us to randomly divide all three sets: 80% for the training set and 20% for the test set. This was done *before* merging the datasets to avoid *data leakage*. Once merged, we will have **3,501 sequences** for the training set and **876** for the test set. Dividing in this way prevents a test set biased toward one of the three starting datasets.
- **Z-Score Normalization and Final Shuffling**: The last step in the pipeline, before running the model with the merged datasets, was to normalize both the terminal set features and the target Y . To do this, we avoided using simple Min-Max scaling, which would have been very sensitive to *outliers* and unusable for values like the MFE (which could have a lower bound of up to -700 kcal/mol). Instead, we opted for **Z-Score Normalization**:

$$z = \frac{x - \mu}{\sigma} \quad (3)$$

Note that we chose to calculate the mean (μ) and standard deviation (σ) using *only* the values from the training set, to simulate the fact that the model does not actually know the values from the test set. This created outliers that were then handled through linearization (*Linear Scaling*), as explained later in this paper. Finally, we randomly permuted (*shuffling*) the final data sets to prevent the model from acquiring biases regarding the position of data belonging to different datasets.

II. METHODOLOGIES

In this section it will be outlined the framework development to predict the mRNA Translation efficiency, it will firstly explain the details behind the synthesis of the biological attributes from the sequences, later used in the Symbolic Regression model described in the implementation details in the subsequent section.

A. Attribute Synthesis

The collected RNA strings cannot be directly plugged into a genetic programming model, because a string of characters don't bring much information and, even by dividing the codons, the expected result wouldn't be usable, and in the form of "2 CAG codons times e^{TCG} " it wouldn't even make much sense. So, being the objective to find a symbolic regression, a terminal set to use must be found. Given all of the characteristics of an RNA strings, 5 macro-characteristics were found, and some of those have been then subdivided.

1) *CG content*: The CG-ratio, or GC-content, indicates the ratio of the nucleotides Guanine (G) and Cytosine (C) bonds in the RNA string, and it impacts the stability and structure of the sequence[6]. Higher CG content leads to stronger hydrogen bonds, making the RNA more stable.

2) *Kozak Strength*: This attribute indicates how well the starting sequence is perceived by the ribosomes[7].

Then, given the list of all the genes in the human body, it's possible to count them individually, building the codon adaptation score, and count the recurrences of the contingent pairs, building the codon-pair matrix. Thanks to these structures, that are fixed once computed, it's possible to get the codon adaptation index and the codon-pair bias.

3) *Codon Adaptation Index (CAI), Ramp and Tail*: The index measures how well a gene's synonymous codon usage matches the preferred version of a specific organism[8]. This index can be then divided in two sub-indexes, that are the ramp and tail CAI. The *ramp* sequence, identified in the initial sequence (20 - 40 codons) of a gene containing less optimal codons (lower CAI) prevent downstream collisions. The remaining *tail* sequence better have higher CAI, for rapid protein synthesis once the ribosome is established.

4) *Codon Pair Bias (CPB)*: Similar to the previous metric, it indicates how well a pair is recognized in the organism - implying better performances during the protein synthetization[9].

By analyzing the folding of the string, it's possible to study the resulting stability[10].

5) *Minimum Free Energy (MFE), Global and Start*: The MFE is a measure of the RNA structural stability, and predicts how tightly the RNA strand folds on itself. The lower (negative) this number is, the stabler it is. This attribute is then divided into two sub-informations: MFE global and start. The first indicates how tight the whole string is, whilst the last indicates how tight the beginning, acting as a "dock" for the conversion process, is. Because of the folding³, the MFE is the computationally heaviest attribute to calculate.

B. Model Description

In order to address our mRNA Translation Efficiency problem, we opted to use symbolic regression. This method searches for a mathematical expression to find an optimal function that approximates the TE efficiency value based on the previously extracted features of an mRNA sequence as input variables for said function.

1) *Implementation Details*: To create our model, we started by implementing the baseline using the *DEAP* (Distributed Evolutionary Algorithms in Python) framework [12]. Here, the population is formed by individuals represented as various syntax trees, where the inner nodes correspond to mathematical operations while the leaf nodes represent the input features or random constants.

2) *Primitive Set*: To create the GP, we had to specify which operations the syntax trees were allowed to use for the interaction between biological features. Many tests were conducted to determine which types of operators fit our case best, involving both linear and non-linear functions, as well as both basic and more complex operations. The main functions used in the most promising tests were the following:

- **Binary Operators**: Addition (+), Subtraction (−), Multiplication (*), Protected Division (/), Power ().
- **Unary Operators**: sin, cos, exp, log, $\sqrt{\cdot}$, absolute value ($|\cdot|$) and negation.

The terminal set (leaf nodes) consists of the extracted features (GC content, ramp CAI, tail CAI, Kozak, CPB, global MFE, start MFE) and Ephemeral Random Constants (ERC) initialized within the range $[-1, 1]$.

3) *Evolutionary Hyperparameters*: In the process of tuning our model for the best performance in exploring the search space, we tested various hyperparameter definitions. For the final model, the configuration was as defined in Table I. We adopted an aggressive approach for the initial search and a contained number of generations, after observing that convergence typically occurred around the 50th generation, with diminishing returns in improvement at higher values.

TABLE I
GENETIC PROGRAMMING HYPERPARAMETERS

Parameter	Value
Framework	DEAP (Python)
Population Size	2000
Generations	100
Crossover probability	0.7
Mutation probability	0.5
Fitness Function	Root Mean Squared Error (RMSE)
Initialization	Ramped Half-and-Half [13]
Selection Method	Tournament Selection
Bloat Control	Tree Height (20-30) and Node Count Limit (200-300)

4) *Linear Scalarization*: The operation that yielded the most significant improvement in the search for the optimal solution was the implementation of Linear Scalarization. A known limitation in classic GP is that it is notoriously hard to fine-tune constants [14]; the algorithm struggles to find coefficients even if the overall structural form of the function is optimal.

In order to address this, we integrated linear scaling within the evaluation loop. This step ensures that each individual is not used directly in its raw version \hat{y}_{raw} ; instead, an Ordinary Least Squares (OLS) regression is performed to find the optimal slope (a) and intercept (b) of said raw output vector that maps the output to the target.

$$\hat{y}_{scaled} = a \cdot \hat{y}_{raw} + b \quad (4)$$

³calculated using the ViennaRNA[11] Package

The fitness metrics (RMSE and Err95%) are then calculated using \hat{y}_{scaled} . This technique provided two major benefits:

- **Structural Focus:** It allows the GP to focus solely on evolving the correct mathematical relationship (correlation) without being penalized for offset or magnitude errors.
- **Improved Convergence:** By removing the burden of constant tuning from the evolutionary process, the algorithm converges significantly faster towards robust solutions.

5) *Fitness Function:* Every individual in the evolutionary process is evaluated based on its performance in the fitness function, which we defined as:

$$Fitness = (0.6 \cdot RMSE + 0.4 \cdot Err95\%) + \lambda \cdot Length \quad (5)$$

This function is a composition of three main components designed to evaluate the precision of the formula created by the syntax tree using the RMSE, ignore the most outlier results to avoid predicting results that would lead the evolution down a wrong path, and lastly, apply a length penalty to prefer the shortest formula at equal performance levels.

- **Root Mean Squared Error (RMSE):** This measures the average magnitude of the error. It is defined as:

$$RMSE = \sqrt{\frac{1}{N} \sum_{i=0}^N (y_i - \hat{y}_i)^2} \quad (6)$$

where y_i is the measured value, \hat{y}_i is the predicted value, and N is the number of samples. We decided to assign this part of the function a weight of 60% of the final result.

- **95th Percentile Error (Err95%):** This metric indicates the value below which 95% of the absolute errors fall. It is included to ensure the model is robust and does not produce extreme outliers; this part was chosen to carry 40% of the weight in the final result.
- $\lambda \cdot \text{Length}$: Acts as a regularizer. Here, **Length** represents the structural complexity of the solution (e.g., the size of the genetic programming tree), and λ is a coefficient that determines the strength of the penalty. This term penalizes overly complex solutions to prevent overfitting and "bloat," favoring parsimony in the final model.

III. RESULTS

The goal of this section is to present the results obtained during the definition of the model and the improvement which each model design decision provided. The model was evaluated by comparing the predicted values against the experimentally measured values on a specific set of mRNA sequences from the test set (see Section I), which were not used during training, in order to also evaluate the generalization capabilities of the model. We used two main metrics for the evaluation: the RMSE (also used in the fitness function) and the Coefficient of Determination (R^2), which assesses how well the model explains the variance.

We observed a significant improvement between the initial approach, where Linear Scalarization was not used, and the later approach which made use of it.

A. Initial Results (Without Linear Scalarization)

Initially, the Genetic Programming model was run without using a Scalarization step, performing the evaluation directly between the raw output of the evolved equation and the target values.

This led to the model generally performing with suboptimal results, achieving an average R^2 score of 0.3. With the best configuration we found, this approach was able to achieve a maximum R^2 score of approximately 0.4, without foreseeable major improvements.

Figure 1 illustrates the correlation between the predicted and actual values for one of the best-performing models from this initial batch ($R^2 \approx 0.3$). The scatter plot reveals how the GP generates a formula that merely scatters the results around a mean, as seen in Example Formula 1 (Eq 1). This indicates a weak linear relationship and a high degree of error in the predictions, suggesting that the raw GP outputs were unable to capture the underlying trend accurately.

$$\left(0.77 - CAI - MFE_S - (0.87^{CAI - MFE_G - CPB}) \cdot (1.27 \cdot GC \cdot (CAI - 0.81) \cdot (MFE_S + 0.66) + (-MFE_S + (KOZ + 0.63)^{0.5}) \cdot \log(CPB - GC))\right)^{0.08} \quad (Eq 1)$$

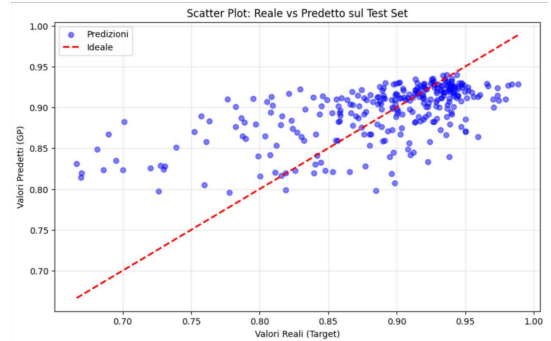


Fig. 1. Predicted vs. Actual values for the best model obtained **without** Linear Scalarization. The points show that the formula of the GP is just predicting values around a certain mean value, resulting in an R^2 of approximately 0.33.

B. Improved Results (With Linear Scalarization)

To deal with the performance limitations in the initial model results, Linear Scalarization (see implementation details in Section II-B4) was applied to the output formulas of the best evolved individuals. This is especially suitable for the kind of problem analyzed here, acting as a linear regression between the raw GP output and the target variable by scaling and shifting the predictions to match the range of the actual data.

This implementation yielded a substantial improvement in predictive accuracy across all trials. The average R^2 score increased by 0.2 to 0.3, giving us an average score of around 0.65. With the optimal configuration identified during the various trials, the best-performing configuration achieved a score of 0.7.

This method led us to formulas like Example Formula 2 (Eq 2), which is concise enough to generalize and does not rely mainly on a single ephemeral constant.

$$GC \cdot tail_{cai} + GC \cdot (kozak - 2.18) + \frac{GC}{kozak} - MFE_{global} - MFE_{start} + tail_{cai} \quad (\text{Eq 2})$$

Figure 2 depicts the correlation for the best model after applying Linear Scalarization ($R^2 \approx 0.69$). In contrast to Figure 1, the data points in this plot cluster more closely around the ideal prediction line. This visual evidence confirms that the application of LS significantly enhanced the model's ability to capture the variance in the experimental data, resulting in much more reliable predictions.

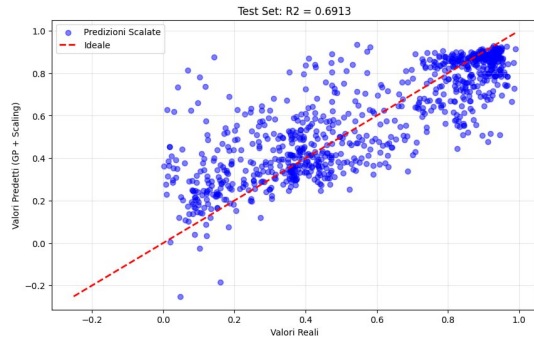


Fig. 2. Predicted vs. Actual values for the best model obtained **with** Linear Scalarization applied. The points are better scattered around the ideal line of prediction values, indicating a stronger correlation and an improved R^2 of approximately 0.69.

C. Other Improvements

Beyond the post-processing steps, additional performance improvements were driven by structural and parameter adjustments within the evolutionary process itself. The implementation of **bloat control** proved effective, preventing the excessive growth of candidate trees without corresponding fitness gains. Furthermore, the complexity of the search space was managed by the **removal of non-linear operators**, steering the evolution toward more robust and interpretable models. Finally, a more **aggressive evolutionary strategy** was adopted; this involved increasing the population size and raising the probabilities for both mutation and crossover, thereby ensuring a broader and more rigorous exploration of the solution space.

IV. CONCLUSIONS

A. Biological Interpretation

Mathematical patterns in the example formula (Eq 2) can be found that align with the biological literature, confirming that the model was able to recognize the importance of the attributes. The term $tail_{cai}$ appears in two different places in the formula, one of which is multiplied by the GC content. This can be explained by the fact that a stable molecule (high GC) has an impact on the TE, especially if it synergizes with

“fast” codons that are easily obtained by the cell ($tail_{cai}$). The negative terms $-MFE_{global} - MFE_{start}$ find a direct correlation, as higher negative MFE values identify more stable structures. The term $GC \cdot (kozak - 2.18)$ could indicate that the model has encountered a sort of translation activation threshold, according to which, below a certain Kozak value, the initiation of the biological process may encounter difficulties. The term of the $GC/kozak$ operation remains the most cryptic and has no direct biological basis; it could indicate that for very weak Kozak values (close to 0), the GC content is the only term capable of shifting the outcome of the TE.

REFERENCES

- [1] S. Li, S. Moayedpour, R. Li, M. L. Kogler-Anele, M. Miladi, J. Miner, F. J. Wang, A. Balsubramani, K. Tran, M. Gu, *et al.*, “Codonbert: Large language models for mrna design,” *arXiv preprint arXiv:2309.12741*, 2023.
- [2] G. Boël, R. Letso, H. Neely, W. N. Price, K.-H. Wong, M. Su, S. Lory, D. Ribet, and J. F. Hunt, “Codon influence on protein expression in *e. coli* correlates with mrna levels,” *Nature*, vol. 529, no. 7586, pp. 358–363, 2016.
- [3] P. J. Sample, B. Wang, D. W. Reid, V. Presnyak, I. J. McFadyen, D. R. Morris, and G. Seelig, “Human 5’ utr design and variant effect prediction from a massively parallel translation assay,” *Nature Biotechnology*, vol. 37, no. 7, pp. 803–809, 2019.
- [4] D. E. Weinberg, P. Shah, S. W. Eichhorn, J. A. Hussmann, J. B. Plotkin, and D. P. Bartel, “Improved ribosome-footprint and mrna measurements boost correlation evaluations for translation efficiency,” *Cell Reports*, vol. 14, no. 7, pp. 1787–1799, 2016.
- [5] R. Wint, A. Salamov, and I. V. Grigoriev, “Kingdom-wide analysis of fungal protein-coding and trna genes reveals conserved patterns of adaptive evolution,” *Molecular Biology and Evolution*, vol. 39, no. 2, p. msab372, 2022.
- [6] G. Kudla, L. Lipinski, F. Caffin, A. Helwak, and P. Collas, “High guanine and cytosine content increases mrna levels in mammalian cells,” *PLoS biology*, vol. 4, no. 6, p. e180, 2006.
- [7] M. Kozak, “An analysis of 5’-noncoding sequences from 699 vertebrate messenger rnas,” *Nucleic acids research*, vol. 15, no. 20, pp. 8125–8148, 1987.
- [8] P. M. Sharp and W.-H. Li, “The codon adaptation index—a measure of directional synonymous codon usage bias, and its potential applications,” *Nucleic acids research*, vol. 15, no. 3, pp. 1281–1295, 1987.
- [9] J. R. Coleman, D. Papamichail, S. Skiena, B. Futcher, E. Wimmer, and S. Mueller, “Virus attenuation by genome-scale changes in codon pair bias,” *Science*, vol. 320, no. 5884, pp. 1784–1787, 2008.
- [10] D. H. Mathews, “Predicting rna secondary structure by free energy minimization,” *Current opinion in structural biology*, vol. 14, no. 3, pp. 285–289, 2004.
- [11] R. Lorenz, S. H. Bernhart, C. Honer zu Siederdissler, H. Tafer, C. Flamm, P. F. Stadler, and I. L. Hofacker, “Viennarna package 2.0,” *Algorithms for Molecular Biology*, vol. 6, pp. 1–14, 2011.
- [12] F.-A. Fortin, F.-M. De Rainville, M.-A. Gardner, M. Parizeau, and C. Gagné, “Deap: Evolutionary algorithms made easy,” *Journal of Machine Learning Research*, vol. 13, no. 70, pp. 2171–2175, 2012.
- [13] J. R. Koza, *Genetic programming: on the programming of computers by means of natural selection*, vol. 1. MIT press, 1992.
- [14] M. Keijzer, “Improving symbolic regression with interval arithmetic and linear scaling,” in *Genetic Programming*, vol. 2610 of *Lecture Notes in Computer Science*, pp. 70–82, Springer, 2003.