A Parametric Study of Interaction-Transformation Evolutionary Algorithm for Symbolic Regression

Guilherme Seidyo Imai Aldeia, Fabrício Olivetti de França
Federal University of ABC
Center of Mathematics, Computing and Cognition
Heuristics, Analysis and Learning Laboratory (HAL)
{guilherme.aldeia, folivetti}@ufabc.edu.br

***Abstract*—The balance between approximation error and model complexity is an important trade-off for Symbolic Regression algorithms. This trade-off is achieved by means of specific operators for *bloat* control, modified operators, limits to the size of the generated expressions and multi-objective optimization. Recently, the representation Interaction-Transformation was introduced with the goal of limiting the search space to simpler expressions, thus avoiding bloating. This representation was used in the context of an Evolutionary Algorithm in order to find concise expressions resulting in small approximation errors competitive with the literature. Particular to this algorithm, two parameters control the complexity of the generated expression. This paper investigates the influence of those parameters w.r.t. the goodness-of-fit. Through some extensive experiments, we find that the maximum number of terms is more important to control goodness-of-fit but also that there is a limit to the extent that increasing its value renders any benefits. Second, the limit to the minimum and maximum value of the exponent has a smaller influence to the results and it can be set to a default value without impacting the final results.**

Keywords: parametric analysis, evolutionary algorithms, symbolic regression.

I. Introduction

Symbolic Regression is the task of finding a closed-form expression that fits a given set of observed samples. The search space explored by these techniques is composed of every possible function-form. This differs from common regression algorithms in which a fixed function-form is provided and only some coefficients are adjusted in order to fit the data.

This task is often performed by a class of evolutionary algorithms (EA) called Genetic Programming (GP). In GP, a solution is represented as an expression tree that describes a valid mathematical expression [1], [2]. As in many EAs, the algorithm starts with a population of solutions that goes through a sequence of recombination, mutation and selection operations until the search converges to a local optima region.

The expressions found by GP resides within the spectrum of *gray-box models* that, unlike the *black-box models*, can be inspected and further analysed [3]. It is often argued that some *black-box* models may be preferred because of their universal approximation properties [4], but the literature shows that GPs can be competitive with many modern regression algorithms [5]–[10].

Even though their performance is competitive, some of these approaches suffer with *bloating* [11]–[13], that results in expressions that resides closer to the *black-box* end of the spectrum. One way to alleviate this problem was proposed in [7] with a new representation for Symbolic Regression, called Interaction-Transformation (IT), that constrains the search space to simpler expressions. The author shows that despite the smaller search space, the quality of the results are still competitive with the literature while keeping the models toward the *white-box* end.

Following the initial success, in [9] the authors proposed a new Evolutionary Algorithm for Symbolic Regression, based on the Interaction-Transformation representation, named Interaction-Transformation Evolutionary Algorithm (ITEA). The obtained results were competitive with the state-of-the-art in Genetic Programming and non-linear regression algorithms.

As with many evolutionary algorithms, ITEA has a set of hyperparameters used to control the balance between exploration and exploitation, and bloat, accuracy, computational time.

In this paper we will analyse two important parameters of ITEA for the symbolic regression task: *max_terms* and *degree_range*. The first one controls the maximum number of interaction terms in an expression and the second one the maximum degree of the generated polynomials.

Also, there is interest in recognizing problem-specific parameters - those who improve significantly the performance on a specific problem - and global parameters - those that can be fixed for most of the cases. The *no free lunch theorem* is also applied here for parameter values, since one configuration can have a good performance on one type of problems, but is likely to perform worse on others [14].

This paper tries to partially answer the following research questions:

- Does there exist a unique set of fixed parameters that frequently dominates other combinations?
- To what extent are goodness-of-fit and size of the model conflicting objectives?
- How the goodness-of-fit varies when we change these parameters?

This paper is organized as follows, in Section II we explain the Interaction-Tranformation representation for Symbolic Re-

978-1-7281-6929-3/20/$31.00 ©2020 IEEE

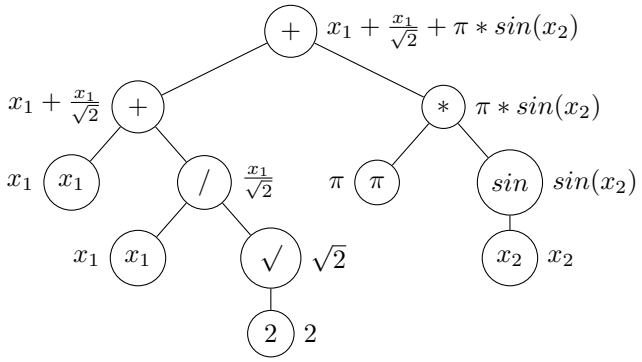

Fig. 1. Function representation as expression tree. On the side of each node it's indicated the sub-function represented by applying its function over their child, and the root represents the final expression.

gression. Following, in Section III we summarize the ITEA algorithm and detail their hyper-parameters. Section IV describes the methodology employed in this paper to answer the proposed research questions. Section V presents the obtained results. Finally, in Section VI we discuss the obtained results and try to answer those questions. Section VII gives some final remarks and points towards new directions given that the questions are answered.

II. Interaction-Transformation representation

Symbolic Regression models describe the relationship between a dependent and independent variables from a data set as a mathematical expression. Given n samples, where each sample has d features, our data set can be represented by $\mathbf{X} = \{\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_n}\}$, and each element of this set represents a d-dimensional vector of features $\mathbf{x_i} = \{x_{i,1}, x_{i,2}, \ldots, x_{i,d}\}$, called explanatory variables; and $\mathbf{y} = \{y_1, y_2, \ldots, y_n\}$ representing the correspondent target variable of each sample. In other words, this regression model searches for a function $\hat{f} : \mathbb{R}^d \to \mathbb{R}$ that minimizes a cost function $J(\hat{f}(\mathbf{X}), \mathbf{y})$. The main advantage of symbolic regression to other machine learning methods is the potential to provide a symbolic function that can be further analysed and interpreted.

Many Symbolic Regression algorithms naturally represent the mathematical expression as *expression trees*, with each internal node being an n-ary function with n child nodes representing their arguments, and the leaves being nodes that represent constant values or one of the explanatory variables. The search for the best expression tree is often performed by an Evolutionary Algorithm, a meta-heuristic that mimics the evolution by means of natural selection over a population of solutions, based on the belief that the selection pressure, mutation and recombination will guide a search that initially appears to be random. Fig. 1 illustrates an expression tree representing the function $f(x_1, x_2) = x1 + \frac{x_1}{\sqrt{2}} + \pi * sin(x_2)$.

While the tree representation is a natural way to represent a mathematical expression, it has some undesired properties that may slow down the convergence of the search algorithm. First of all, when an expression is changed through mutation or when two expressions are combined through crossover, the resulting expression may have a very different semantic in comparison to the parents expression. This problem is often alleviated by means of semantic operators [12], [15].

Another property is that this representation contains the entirety of possible mathematical expressions in its search space. This means that the search space contains redundant expressions and very complex models that can be considered black-box models [7].

In [7], the author proposed a new representation that alleviate some of these problems by reducing the search space with the cost of possibly having a chance of removing the global optima from it. The main inspiration was to create a restricted grammar that could represent many engineering and physics equations.

Given a unary function $t : \mathbb{R} \to \mathbb{R}$, called transformation function and a d-dimensional interaction function p, $p : \mathbb{R}^d \to \mathbb{R}$, we can write the composition $t \circ p$ representing the transformation function applied to the interaction function, therefore called an Interaction-Transformation term.

The interaction function is created by providing a vector $\mathbf{k} \in \mathbb{Z}^d$ of exponents, called strengths, in order to create a polynomial interaction function:

$$p(\mathbf{x}) = \prod_{i=1}^{d} x_i^{k_i}. \tag{1}$$

An Interaction-Transformation expression is then defined as the linear combination of multiple IT terms:

$$\widehat{f}(\mathbf{x}) = w_0 + \sum_{j=1}^{m} w_j \cdot (t_j \circ p_j)(x), \tag{2}$$

where w_j is the j-th coefficient of the linear combination of interaction-transformation terms and m is the number of terms.

III. Interaction-Transformation Evolutionary Algorithm

Following the successful results of the IT-based greedy algorithm proposed in [7], in [9] the authors proposed a mutation-based evolutionary algorithm called Interaction-Transformation Evolutionary Algorithm (ITEA) [9], summarized in algorithm 1.

Algorithm 1: Interaction-Transformation Evolutionary Algorithm

input : data points $\mathbf{X}, \mathbf{y}$
output: Symbolic function $\widehat{f}$

pop ← GenRandom();
while *criteria not met* **do**
  children ← [Mutate(p) for $p \in$ pop];
  pop ← Select(pop + children);
return $arg\ max_{p.fit}\ pop$

The algorithm starts with a randomly generated population of solutions. After that, it performs the mutation and selection steps repeatedly. The *mutation* operator slightly modifies each solution in the population and the *selection* operator chooses the next population with tournament selection. In the implementation proposed in [9], there are five possible mutations, with equal chance of being selected: adding a new term, dropping a term from the expression, positive and negative interactions of two terms, replace one strength coefficient of one interaction.

Adding a term simply adds a new term to the expression, while dropping a term removes a randomly chosen term from the expression. In Positive and Negative interactions, the algorithm selects two terms at random and then adds/subtracts their exponents replacing the values of one of those terms and, finally, replacing an interaction simply replaces a chosen exponent of a random term.

During the evaluation of an individual solution, every term that evaluates to NaN (*Not a Number*, indicating that a invalid mathematical operation was made (i.e. dividing by zero) or the result is to large to be expressed) is removed from the expression. So the fitness evaluation changes the expression not only by adjusting the weights of the linear regression but by removing invalid terms. Because of this mechanism, it is impossible to create an expression that evaluates to NaN for the training data.

The behavior of this algorithm is controlled by a set of 7 user-defined parameters. Common to GP literature, we have the population size, stop criteria and functions set. Exclusive to this algorithm, the user must also choose the maximum number of terms on an IT-expression, the degree range for the exponents, the fitting algorithm for the linear model, and the fitness function.

For this paper we will limit the study of the parameters regarding the number of IT terms (max_terms) and the exponent limits ($degree\_range$), since they explicitly controls the level of complexity of the IT-expression following a criteria that the smaller the expression the easier it becomes to understand it.

IV. Methodology

This section describes the methodology followed to estimate the effect of the two studied parameters.

To assess how these parameters can impact the overall performance, we determined a range of values for each of them and generated all the combinations for those values, as depicted in Table I. Notice that a $degree\_range$ value of x defines a range of allowed exponents as a closed interval $[-x, x]$.

Small values for those two parameters restricts the search space for simpler expressions that may be simpler to understand and analyze. On the other hand, if the search space is too restrictive it may not be possible to adequately fit a given data set.

TABLE I
SET OF HYPERPARAMETERS AND THEIR VALUES. THE TOP PART OF THE TABLE DESCRIBES THE RANGE OF VALUES CONSIDERED IN THIS STUDY, THE BOTTOM PART DESCRIBES THE FIXED VALUES.

Parameter	Values	Description
$degree\_range$	[1, 2, 3, 4, 5]	The upper and lower limits to exponents
max_terms	[2, 4, 6, 8, 10]	The max number of IT terms allowed in a IT expression
$gens$	300	The number of generations
pop	100	Population size
f_set	$[\sin, \cos, \tanh, \sqrt{\lvert . \rvert}, \log, \exp]$	Transformation functions
$solver$	OLS	Linear Regression solver

TABLE II
NUMBER OF FEATURES, SAMPLES AND RATIO OF TRAINING AND TEST SAMPLES FOR EACH DATASET.

Name	# features	# samples (train/test)
Airfoil	6	1202/301
Concrete	9	824/206
Energy Cooling	9	614/154
Energy Heating	9	614/154
Tower Data	26	3999/1000
Wine Red	12	1279/319
Wine White	12	3919/979
Yacht	7	246/62

Regarding the remaining parameters, they were all fixed with the values also depicted in Table I, and will be the subject of study in future works.

In order to measure the influence of these parameters, a set of commonly used data sets (see Table II) were chosen and split into a 5-fold setup for cross-validation. For every dataset we have repeated the execution of ITEA for 30 times with each combination of values of $degree\_range$, max_terms. The performance of each experiment was measured using the Root Mean Squared Error (RMSE), the same as the fitness function. The code, test scripts and raw data can be found at https://github.com/gAldeia/sensitivity-analysis-ITSR.

In the next section we will report the median of the RMSE and the ranks obtained by each configuration of hyperparameters. These two measures might differ in situations that one configuration have a better average performance with a high variance.

Additionally, we also measure the importance of each parameter with the Coefficient of Variation. Given a parameter tuning problem with n parameters, we use the coefficient of variation to estimate the impact of changing the value of each parameter. The coefficient of variation, denoted by C_v, is a measure of dispersion expressed by a percentage obtained by dividing the standard deviation to the mean of a given distribution.

The standard deviation is a good measure of dispersion but it cannot be directly compared to the standard deviation of another distribution with different scale. Instead, by using C_v,

we can perform this comparison, because they are measured on the same relative scale. It can also be used to estimate the stability of individual parameters, where a high C_v stands for a less stable variable, and a low C_v for a more stable variable.

Given a set of parameters P, with each $p_i \in P$ having a domain of possible (or considered) values $v_j \in V_i$, we can calculate the coefficient of variation $C_v^{p_i}$ for a parameter p_i with an algorithmic procedure. For every combination c_k of values of the remaining parameters set $P - \{p_i\}$ we perform 30 repetitions of the algorithm for each value $v_j \in V_i$ and store the median of the obtained RMSE. We then calculate the mean and standard-deviation of these median values, calculating

$$C_v^{p_i|v_j} = \bar{x}/std(x), \tag{3}$$

where x is the set of medians calculated by each experiment. Finally, we can calculate $C_v^{p_i}$ as:

$$C_v^{p_i} = \frac{1}{|V_i|} \sum_j C_v^{p_i|v_j}. \tag{4}$$

V. Results

The median of the RMSE values of the best individual applied to the test set for each configuration is reported in Fig. 2. Since we are analyzing only two parameters, it is possible to visualize the results as a heatmap representing a discrete landscape of the hyperparameters performance.

In this plot the darker colors represent the smaller values for RMSE, thus representing the best obtained results. Additionally, the best result is underlined and all the results with p-value > 0.05 calculated with the Wilcoxon Rank-Sum test compared against the best result is marked with an asterisk. Notice also that the results with equal values but different colors differs on the third decimal places onward.

For every dataset, with the exception of Wine Red, the horizontal gradient of colors is smooth when observing the x-axis representing max_terms values. This means that, in those datasets, maximizing the number of terms help to reduce the generalization error. Specifically speaking of the Wine Red dataset [16], the target variable represents the quality of a given wine on a scale of 1 to 10. Most of the wines are rated within an average range and just a few are excellent or poor. Our hypothesis is that this imbalance make this dataset prone to overfitting and, thus, simpler models are preferred.

Regarding $degree\_range$, we can see that in every configuration of $max\_terms = 10$, either $degree\_range$ value of 3 was the best result or it was close to the best. The only exception to this rule is for the Yacht data set in which the higher this value the better the results, this may happen when the underlying system does not reside inside the IT search space, thus leading to an approximation with higher degree polynomials. In short, the value for $degree\_range$ should be set according to some prior knowledge of the studied system or, if that is not a possibility, using a hyperparameter tuning technique.

TABLE III
Average percentage of decrease in test data RMSE when increasing the parameter max_terms.

Dataset	$2-4$	$4-6$	$6-8$	$8-10$
Airfoil	19.06%	7.06%	5.85%	5.38%
Concrete	12.22%	6.28%	1.74%	2.35%
Energy Cooling	29.51%	16.17%	7.95%	1.64%
Energy Heating	37.35%	42.84%	27.36%	11.82%
Tower Data	8.59%	5.97%	4.19%	2.82%
Wine Red	2.05%	0.71%	−0.77%	0.21%
Wine White	1.74%	1.05%	0.55%	−0.02%
Yacht	18.75%	13.47%	11.75%	4.11%

On the other hand, the value for max_terms should be chosen as high as the practitioner judges reasonable in order to keep the returned expression within the *gray-box* region. Even though the higher the value of this parameter the smaller the approximation error, the difference between similar values of max_terms decrease as the value for this parameter increases.

In Table III we can see the average percentage of decrease in the RMSE for the test data when we increase the value of max_terms. Each column of this table calculates this percentage from two adjacent values of max_terms as

$$P_{dec}(rmse1, rmse2) = 100 \times \frac{rmse1 - rmse2}{rmse1}. \tag{5}$$

From this table it is possible to see that the percentage of decrease becomes smaller as max_terms is set to a higher value. This indicates that there can be a threshold for this parameter in which the increase of complexity will lead to a negligible decrease in approximation error. Following this result, we can investigate the use of an adaptive value for this parameter throughout the evolution.

The median value of the error measure is commonly used in order to alleviate the occurrence of outliers when performing experiments with stochastic algorithms. But, sometimes we are not interested on the best average result but on those results that are better ranked on average. Fig. 3 depicts the average rank of each configuration for each data set. The results of this plot may differ from those depicted on Fig. 2 when a given configuration is ranked first for most of the folds of the data set but in some folds it gets a very low rank.

As we can see from this plot, the results follow the same trend as the previous plot w.r.t. max_terms, the higher the value of this parameter the better the rank. But, for $degree\_range$ the new optimal value is shifted to $degree\_range = 2$, indicating that a lower degree is capable of maintaining a better stability throughout the folds.

In Fig. 4 we show a convergence analysis for each configuration of the parameters values. This plot shows the average RMSE over the 30 executions of each configuration on every generation of ITEA applied to the test set. The worst configuration w.r.t. the final RMSE is highlighted in red and the best one in blue. The values of the parameters for both

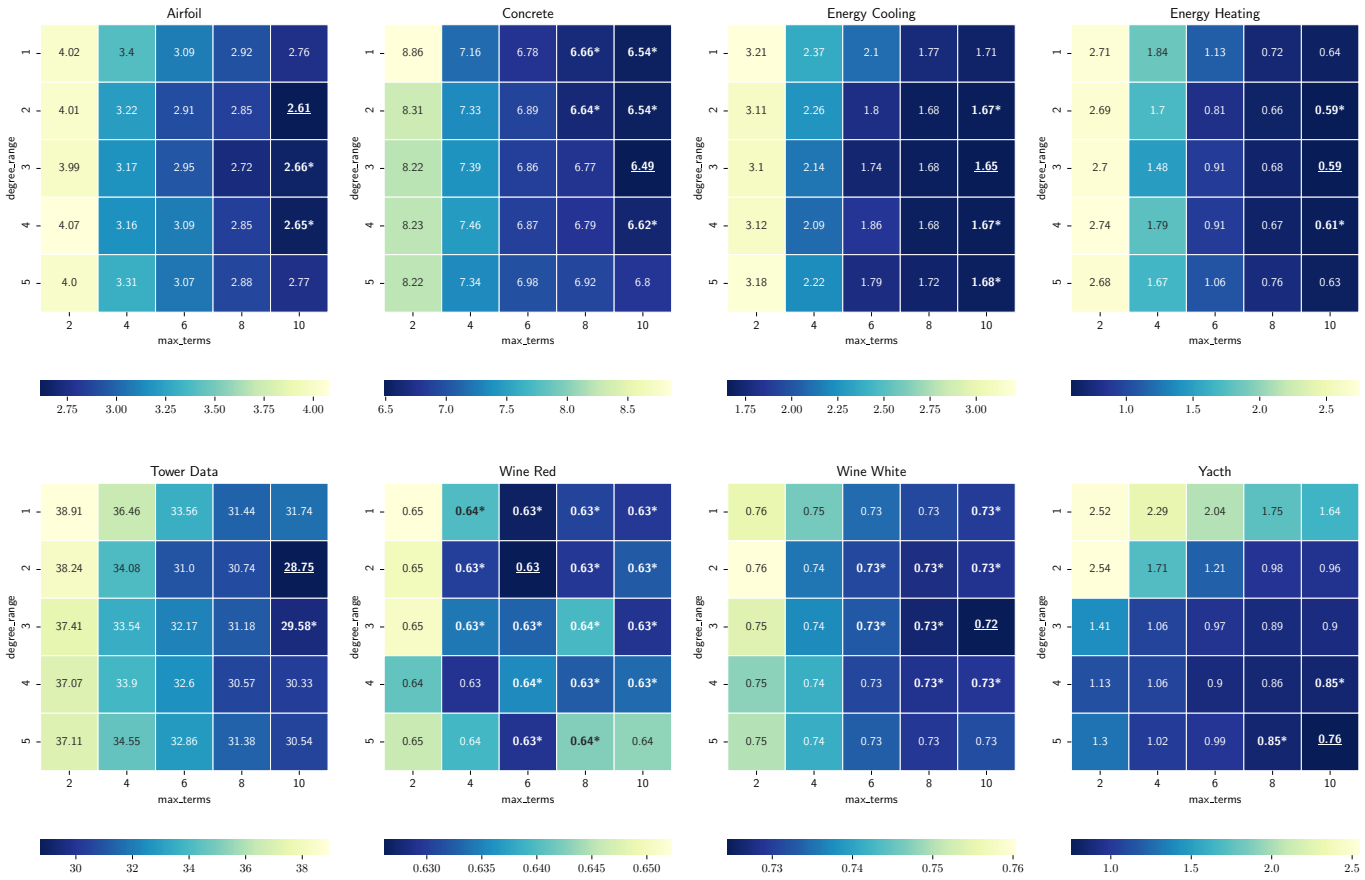

Fig. 2. Heatmap showing the median RMSE values over 30 runs for each data set. The smaller the value, the better the results. The best result is underlined and every result with a p-value > 0.05 when compared to the best result is marked with an asterisk.

configurations are annotated in the plot. Notice that in some rare occasions for the *concrete* data set, the initial fitness of the individuals evaluated to *NaN* **for the test data**, these points are not plotted in the graph.

The behaviors observed on these plots can be divided into three different groups. For the first group comprehending the data sets *Airfoil, Wine Red, Wine White, Energy Heating* we can see that the worst configuration converges prematurely, usually before 50 generations, and remains stable for the entirety of the execution. Meanwhile, the best configuration shows a behavior of improvement until the final generation, even though the rate of improvement is greatly reduced after the 100th generation.

The second group composed of the data sets *Energy Cooling, Yacht* shows that both the best and worst configurations converges much before the final generation. The final group, composed only of the *Tower Data* shows that both the best and worst configuration still have not converged after the 300th generation, it should be noted that this data set has the most number of features and, thus, it should take longer to converge.

Because of the NaN problem, the convergence for the *Concrete* data set cannot be analysed. But, we can notice that, even though the first generations contained NaN values, the best configuration still manages to find a proper solution after the 150th generation. Notice that, while it is impossible to obtain a NaN value for the training set, since the invalid terms are discarded, it can be the case that the best expression is invalid when extrapolating. This could be alleviated with the inclusion of prior knowledge about the data [17].

Overall, we can also notice that the worst configurations are usually the combination of the smaller values for each parameter.

Tables IV and V shows the coefficients of variation for $degree\_range$ and max_terms, respectively. The value C_v^i correspond to the coefficient of variation for the parameter value i, $\overline{C_v}$ is the average coefficient of this parameter to the corresponding data set.

From these tables we can corroborate what was already observed from the plots. The average coefficients for $degree\_range$ is smaller than 5.8%, meaning that we can expect this variation to the RMSE when we change this parameter. The only exception being for the *Yacht* data set in which a rate of variation of 33.5% was observed. Regarding max_terms, the measured variations range from 1.1% to 59%. The most impact of this parameter was observed for the *Energy Heating, Energy Cooling, Yacht* and *Airfoil*, in

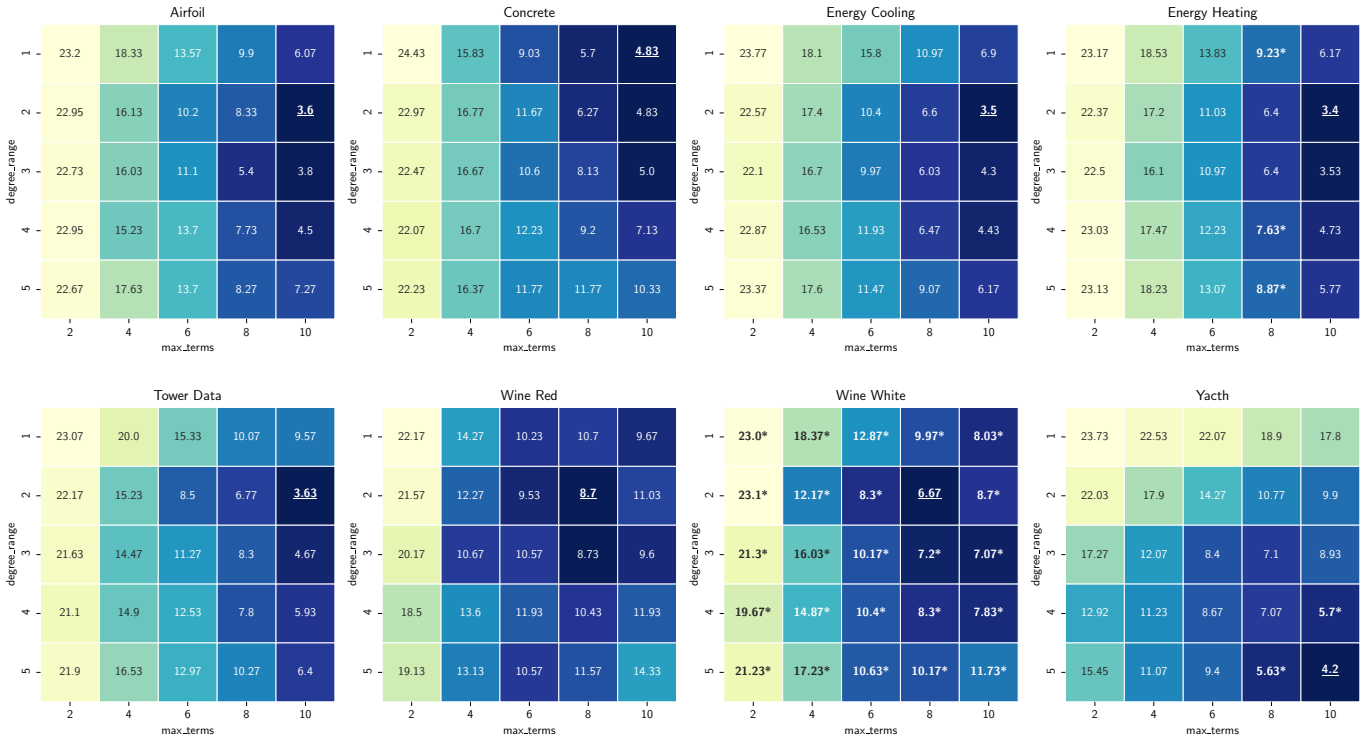

Fig. 3. Heatmap showing the average ranks over 30 runs for each data set. The smaller the value, the better the results. The best result is underlined and every result with a p-value > 0.05 when compared to the best result is marked with an asterisk.

TABLE IV
TABLE OF C_vS FOR THE $degree\_range$, WHERE EACH C_v^i IS OBTAINED BY FIXING max_terms AND TESTING WITH ALL POSSIBLE VALUES FOR $degree\_range$.

Dataset	C_v^1	C_v^2	C_v^3	C_v^4	C_v^5	$\overline{C_v}$
Airfoil	0.7%	2.8%	2.6%	2.3%	2.3%	$2.1\% \pm 0.7$
Concrete	3.0%	1.3%	1.0%	1.5%	1.7%	$1.7\% \pm 0.7$
Energy Cooling	1.4%	4.4%	6.9%	2.0%	1.2%	$3.2\% \pm 2.2$
Energy Heating	0.8%	7.4%	12.1%	5.1%	3.6%	$5.8\% \pm 3.8$
Tower Data	1.9%	3.0%	2.6%	1.1%	3.3%	$2.4\% \pm 0.8$
Wine Red	0.6%	0.7%	0.6%	0.8%	0.8%	$0.7\% \pm 0.1$
Wine White	0.6%	0.5%	0.2%	0.2%	0.3%	$0.4\% \pm 0.2$
Yacht	34.8%	35.0%	34.5%	32.1%	31.0%	$33.5\% \pm 1.6$

TABLE V
TABLE OF C_vS FOR THE max_terms, WHERE EACH C_v^i IS OBTAINED BY FIXING $degree\_range$,AND TESTING WITH ALL POSSIBLE VALUES FOR max_terms.

Dataset	C_v^2	C_v^4	C_v^6	C_v^8	C_v^{10}	$\overline{C_v}$
Airfoil	13.7%	15.6%	15.5%	15.5%	13.7%	$14.8\% \pm 0.9$
Concrete	11.9%	9.0%	8.6%	8.2%	7.1%	$9.0\% \pm 1.6$
Energy Cooling	24.4%	26.1%	26.6%	25.8%	26.7%	$25.9\% \pm 0.8$
Energy Heating	55.2%	62.3%	61.3%	60.8%	55.3%	$59.0\% \pm 3.1$
Tower Data	8.3%	10.2%	8.1%	7.5%	7.1%	$8.2\% \pm 1.1$
Wine Red	1.5%	1.3%	1.2%	0.6%	0.9%	$1.1\% \pm 0.3$
Wine White	1.3%	1.6%	1.4%	1.0%	1.0%	$1.3\% \pm 0.2$
Yacht	16.0%	40.2%	18.3%	11.9%	18.7%	$21.0\% \pm 9.9$

this order. An impact of less than 9% was observed on the remainder of the data sets

Finally, in Table VI we depict sample expressions obtained by different configurations of $\{degree\_range, max\_terms\}$ for the *Yacht* data set. From this table we can see the compromise between approximation error and model complexity. While the expression with $max\_terms = 10$ is still far from a *black-box* model, the alternative with only two terms and maximum exponent of 4 is much easier to analyze without compromising too much the approximation error.

VI. DISCUSSIONS

In the beginning of this paper, we posed three questions we wanted to answer with these experiments (see Section I):

- **Does a set of fixed settings that frequently dominate the others exist?** None of the studied parameters showed a common C_v value, neither the same *landscape* was observed between all heatmap plots. This implies that the choice of parameters is problem-specific. The only observed pattern was that, in most cases, a higher max_terms leads to a smaller approximation error. Setting a value sufficiently high for this parameter would reduce the problem to adjusting the value of only one parameter.
- **To what extent are goodness-of-fit and interpretability conflicting objectives?** The most important parameter for interpretability is max_terms, which increases the size of the expression while decreasing the error (see previous

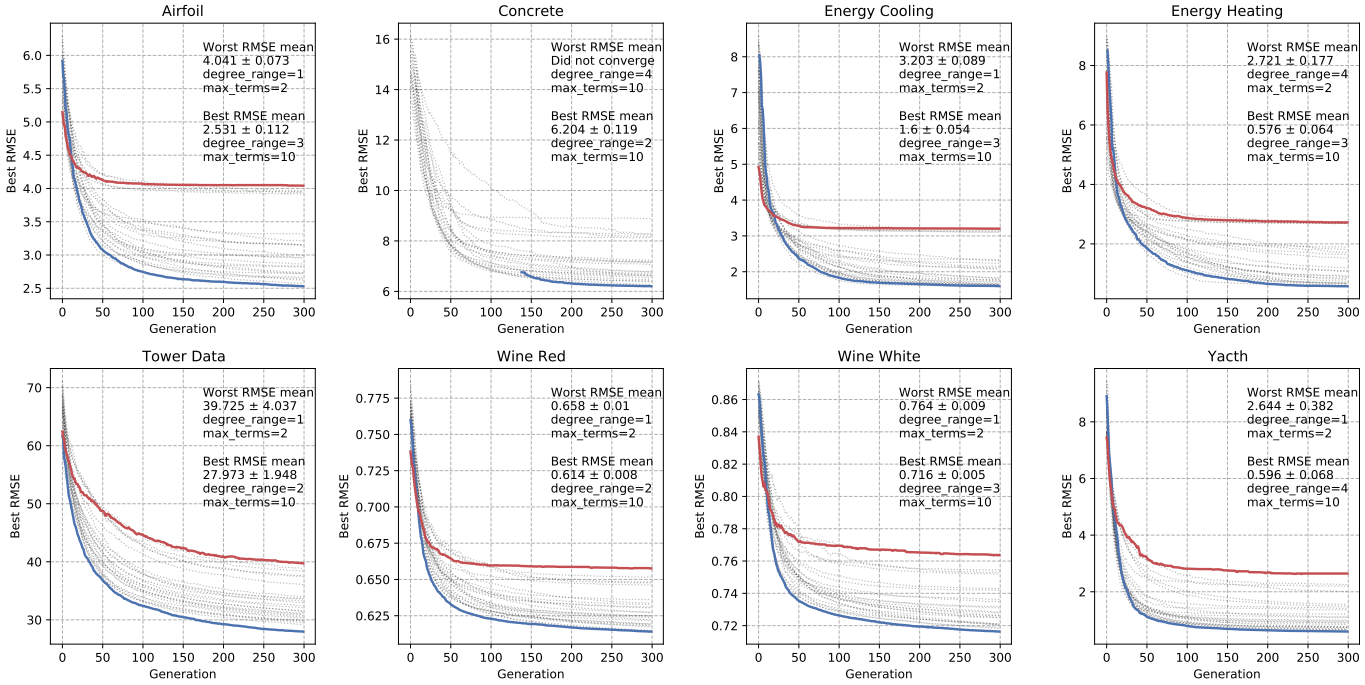

Fig. 4. Average RMSE of the best individual obtained at each generation for each parameters configuration. The best (blue) and the worst (red) configurations are emphasized.

TABLE VI
EXAMPLE OF EXPRESSIONS OBTAINED WITH DIFFERENT CONFIGURATIONS FOR YACHT. THE FIRST COLUMN DEPICTS THE HYPERPARAMETERS VALUES AS A TUPLE $(degree\_range, max\_terms)$.

Conf.	RMSE	Expression
$(1, 2)$	2.21	$-2.83e+03 \tanh\left(\frac{1}{x5}\right) + 6.63e+02 \cos(x1 \cdot x5) + 2.17e+03$
$(4, 2)$	0.79	$-7.79e+03 \cos\left(\frac{x2^3 \cdot x5^4}{x3 \cdot x4^2}\right) - 4.05e+02 \cos\left(\frac{x1^4 \cdot x2^3 \cdot x5}{x3 \cdot x4^2}\right) + 8.19e+03$
$(5, 10)$	0.55	$5.72e+05 \tanh\left(\frac{x1}{x5^3}\right) + 6.44e+06 \tanh\left(\frac{1}{x0 \cdot x1^4 \cdot x5^2}\right) + 1.60e+08 \cos\left(\frac{x5^2}{x2^3 \cdot x3^2}\right) - 5.75e+06 \cos\left(\frac{x5^4}{x1^2 \cdot x2^2 \cdot x3}\right) - 5.61e+04 \cos(x1^3 \cdot x5^3) + 8.00e+02 \tanh(x1 \cdot x5^4) - 1.29e+07 \tanh\left(\frac{x0 \cdot x1^3 \cdot x4}{x5^5}\right) - 3.81e+04 \tanh\left(\frac{x1}{x5^2}\right) + 1.23e+04 \tanh\left(\frac{x1^4 \cdot x6^4}{x0 \cdot x2}\right) - 1.61e+08$

answer). It is often the case that reducing the expression complexity by adjusting the value of max_terms will increase the approximation error, thus showing that they are indeed conflicting objectives. One thing to notice, though, is that the percentage of decrease of the RMSE becomes smaller at every increase of max_terms value (see Table III), so there seems to have a point in which there is no benefit in increasing the expression complexity any further.

- **How the goodness-of-fit varies when we change each parameter?** Based on the observed, given the range of values tested, the max_terms is more important to achieve a smaller error, though it can lead to a decrease on the interpretability of the expression (see previous answer). The $degree\_range$ choice presented a minor variation and should be either chosen by using prior knowledge about the data or by tuning through a hyperparameter optimization technique.

VII. CONCLUSION

In this paper we analysed the relevance of two parameters of the Interaction-Transformation Evolutionary Algorithm with respect to the approximation error (RMSE) and model complexity.

For this purpose we have setup a series of experiments with different data sets revolving around the tuning of two parameters: $degree\_range$ and max_terms. We have performed a grid search over 5 different values for each parameter with repeated experiments due to the stochastic nature of the algorithm.

We have then analysed two Heatmap plots, one for the RMSE on the test set and another for the average rank of each configuration. Additionally, we have calculated the coefficient of variation of each parameter in order to measure how much

the tuning of each parameter affects the performance of the algorithm.

From these analysis we could answer three research questions: while there is no configuration that greatly dominates all other, we can safely focus our attention to fine-tuning only max_terms. The only exception to this rule was observed in *Yacht* data set, in which the variation of both parameters were important. Another answered question is that the goodness-of-fit and model complexity are two conflicting objectives, so if you want a fittest model for your data you should expect an increase in its complexity. On the other hand, the benefits of increasing the complexity quickly decreases over the studied values of max_terms.

As a novelty to the study of hyper-parameters, we have calculated the Coefficient of Variation of each parameter. This coefficient determines a percentage of how much the RMSE varies when we change the value of a single parameter. The obtained values for this coefficient showed us that max_terms is the most important of the two studied parameters w.r.t decreasing RMSE. The results corroborates with the Heatmap plots and answers the final research question posed on this paper.

Additionally, the convergence plots of each configuration for each data set was presented so we could see that due to the larger search space induced by a higher value of max_terms, the convergence rate is usually much slower than for smaller values of this parameter.

Overall, the analysis of the Heatmap plots and Coefficients of Variation can highlight the importance of each parameter, in a way that is possible to determine the impact on the performance for different combinations of values for these parameters.

ACKNOWLEDGMENT

This project was funded by Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP), grant number – 2018/14173-8.

REFERENCES

[1] John R Koza. Genetic programming. 1997.

[2] William B Langdon and Riccardo Poli. *Foundations of genetic programming*. Springer Science & Business Media, 2013.

[3] Lennart Ljung. Perspectives on system identification. *Annual Reviews in Control*, 34(1):1–12, 2010.

[4] Jooyoung Park and Irwin W Sandberg. Universal approximation using radial-basis-function networks. *Neural computation*, 3(2):246–257, 1991.

[5] Ignacio Arnaldo, Una-May O'Reilly, and Kalyan Veeramachaneni. Building predictive models via feature synthesis. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, pages 983–990. ACM, 2015.

[6] William La Cava, Tilak Raj Singh, James Taggart, Srinivas Suri, and Jason H Moore. Learning concise representations for regression by evolving networks of trees. 2018.

[7] Fabrício Olivetti de França. A greedy search tree heuristic for symbolic regression. *Information Sciences*, 442:18–32, 2018.

[8] Guilherme Seidyo Imai Aldeia and Fabrício Olivetti de Franca. Lightweight symbolic regression with the interaction-transformation representation. In *2018 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8. IEEE, 2018.

[9] Fabricio Olivetti de Franca and Guilherme Seidyo Imai Aldeia. Interaction-transformation evolutionary algorithm for symbolic regression, 2019.

[10] Michael Kommenda, Bogdan Burlacu, Gabriel Kronberger, and Michael Affenzeller. Parameter identification for symbolic regression using nonlinear least squares. *Genetic Programming and Evolvable Machines*, pages 1–31.

[11] William B Langdon and Riccardo Poli. Fitness causes bloat. In *Soft Computing in Engineering Design and Manufacturing*, pages 13–22. Springer, 1998.

[12] Joao Francisco Martins, Luiz Otavio VB Oliveira, Luis F Miranda, Felipe Casadei, and Gisele L Pappa. Solving the exponential growth of symbolic regression trees in geometric semantic genetic programming. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1151–1158. ACM, 2018.

[13] João Victor C Fracasso and Fernando J Von Zuben. Multi-objective semantic mutation for genetic programming. In *2018 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8. IEEE, 2018.

[14] S. K. Smit and A. E. Eiben. Parameter tuning of evolutionary algorithms: Generalist vs. specialist. In *Applications of Evolutionary Computation*, pages 542–551, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[15] Tomasz P Pawlak. Competent algorithms for geometric semantic genetic programming. *review. PhD thesis. Pozna'n, Poland: Poznan University of Technology*, 2015.

[16] Paulo Cortez, António Cerdeira, Fernando Almeida, Telmo Matos, and José Reis. Modeling wine preferences by data mining from physicochemical properties. *Decision Support Systems*, 47(4):547–553, 2009.

[17] Stefan Prieschl, Dominic Girardi, and Gabriel Kronberger. Using ontologies to express prior knowledge for genetic programming. In *International Cross-Domain Conference for Machine Learning and Knowledge Extraction*, pages 362–376. Springer, 2019.