Tuning Genetic Programming Parameters with Factorial Designs

Elisa Boari de Lima, Gisele L. Pappa, Jussara Marques de Almeida,
Marcos A. Gonçalves and Wagner Meira Júnior

Abstract— **Parameter setting of Evolutionary Algorithms is a time consuming task with two main approaches: parameter tuning and parameter control. In this work we describe a new methodology for tuning parameters of Genetic Programming algorithms using factorial designs, one-factor designs and multiple linear regression. Our experiments show that factorial designs can be used to determine which parameters have the largest effect on the algorithm's performance. This way, parameter setting efforts can focus on them, largely reducing the parameter search space. Two classical GP problems were studied, with six parameters for the first and seven for the second problem. The results show the maximum tree depth as the parameter with the largest effect on both problems. A one-factor design was performed to fine-tune tree depth on one problem and a multiple linear regression to fine-tine tree depth and number of generations on the other problem.**

I. Introduction

Although the success of evolutionary algorithms (EAs) in a vast number of applications is certainly related to a good choice of individual representation and fitness, the best choice of these two can still lead to poor results if the algorithm's parameters are not carefully set. Making parameter setting a less time consuming, application-driven and ad hoc process [1], [2] is one of the priorities of the EA community.

In the EA parameter setting literature, there are two main approaches to be followed: parameter tuning and parameter control [1]. The main difference among them is that while parameter control is done online (i.e., during the execution of the EA), parameter tuning is made a priori, with the parameters remaining static during the EA evolution [3]. This work focuses on the latter approach and describes a new methodology for tuning parameters of a genetic programming algorithm.

Parameter tuning is a hard problem because there is a great number of possible parameter value combinations, which have to be set with very little available information regarding the effect that each parameter has on the EA performance [4]. Most EA users rely on rules of thumb (e.g. mutation rates should be low) and ad hoc choices of parameters [5]. However, in recent years, a few methods for making this task more systematic have been proposed. Among them are the works of Czarn et al. [6], which use statistical tests to find the degree of interaction between crossover and mutation rates, and the use of (Meta) Estimation of Distribution Algorithms (EDAs) [7] and (Meta) Genetic Algorithms [2] for parameter tuning. The majority of these methods focus on tuning parameters for genetic algorithms, and search the space of parameter values using some sort of heuristic.

The authors are with the Department of Computer Science, ICEx, Universidade Federal de Minas Gerais, 31270-010 Belo Horizonte, Minas Gerais, Brazil (emails: {eblima, glpappa, jussara, mgoncalv, meira}@dcc.ufmg.br).

Many disciplines in Computer Science and Engineering require parameter setting. Factorial experimental designs [8] are one of the most broadly applied techniques for this purpose. The idea behind a factorial design is to obtain the maximum information about the factors (i.e., parameters) with the minimum number of experiments. Furthermore, it allows us to determine whether a parameter's effect on the algorithm's output is significant or if the observed difference among executions with different parameter values is simply due to random variations caused by measurement errors and/or uncontrolled parameters. The methodology for parameter tuning here proposed relies on a 2^k factorial design to estimate the influence of the parameters on the GP runs. Once the parameters with the largest effects on the GP's output are identified, we focus on fine tuning them using regression techniques.

Feldt & Nordin [9] have used the *fractional factorial design* to analyze a machine code genetic programming system. The main difference between a fractional factorial design and the full factorial design applied in our proposed methodology is that, in the former, the effects of independent parameters are always confounded with the effect of a parameter interaction, while the latter is capable of isolating the effects of both parameter interactions and independent parameters. Moreover, Feldt & Nordin [9] analyzed sixteen factors, including the constants used and the presence or absence of a function in the function set. This work focuses on six basic GP parameters while using a (full) 2^k factorial design: number of generations, population size, tree depth, and crossover, mutation and reproduction rates.

Note that the main goal of the proposed methodology is not to reduce the number of experiments performed, but to measure the effect each parameter has on the problem being tacked by using two "extreme" parameter values. The parameters identified as the ones with the largest impacts on the fitness values can then be given as input to other parameter tuning methods previously proposed in the literature, or refined using the approaches proposed in this work.

The results obtained show the maximum tree depth as the parameter with the largest effect on the two investigated problems. Hence, a one-factor design was performed to fine-tune tree depth in the symbolic regression problem. For the 11-multiplexer problem, the number of generations also presents a large effect, so a multiple linear regression was performed to fine-tune both parameters.

The remainder of this paper is organized as follows. Section II reviews previous work on parameter tuning, while

Section III describes the basic approach of the 2^k factorial design. Section IV describes the setup used for the experiments, which focused on two problems (the symbolic regression of function $x^4 + x^3 + x^2 + x$ and the boolean 11-multiplexer problem), and discusses the results. Finally, Section V draws some conclusions and directions for future work.

II. Related Work

According to [1], there are two main approaches to setting parameter values: *parameter tuning* and *parameter control*. The former consists of finding good values for the parameters before running the algorithm and then using these fixed values to run the algorithm, whereas the latter starts running the algorithm with an initial parameter setting that changes during the algorithm's execution according to a control strategy which can be deterministic, adaptive or self-adaptive.

Parameter tuning is accomplished by experimenting with different values and selecting the ones that give the best results on the test problems at hand. This is very time-consuming given the possibly explosive number of parameters and parameter values. EA users mostly rely on conventions, ad hoc choices and experimental comparisons on a limited scale for parameter tuning. There is no tuning algorithm widely used by the community. Therefore, a large, systematic investigation of parameter effects using Genetic Programming would enhance experiments by providing insights on the behavior of parameters in specific applications.

The work most closely related to ours is that of Feldt & Nordin [9], which is the first series of experiments addressing parameter influence in a broad and systematic way. The authors apply statistical techniques for designing and analyzing experiments in order to evaluate the individual and combined effects of GP parameters. The experimental method used in their work is a *fractional factorial design*, which was chosen for its ability to reduce the number of runs that are needed and to increase the amount of knowledge that can be gained. However, despite saving costs when compared to full factorial designs, according to Jain [8] the information obtained from a fractional factorial design is less than that obtained from a full design since the main parameter effects are never "pure", but always confounded with the effect of parameter interactions. Moreover, fractional factorial designs rely on the supposition that the effect of a parameter interaction is negligible, which might not be true.

Feldt & Nordin [9] studied three standard machine learning binary classification problems, and considered a total of sixteen parameters. The authors used, in sequence, three fractional factorial designs of increasing resolution (the third design was applied to only one of the problems), and concluded that on the three problems population size and number of generations were the most significant parameters. The results also showed that mutation and crossover probabilities had positive effects on one problem, but their effects are not significant on the other two. The authors also noted that there were significant two-factor interactions in the latter cases. This fact suggests that the fractional factorial design is inappropriate for GP experimental design, since the effect of different two-factor interactions cannot be separated with the resolution they adopted.

Smit & Eiben [2] investigated the feasibility of using parameter tuning algorithms, describing three approaches to algorithmic parameter tuning: meta-EA, meta-EDA and SPO. The analyzed tuning system is composed of three layers, namely application layer, algorithm layer and design layer. The design method attempts to find optimal parameters for the EA on the algorithm layer, which tries to find an optimal solution for the problem on the application layer. The authors conclude that using algorithms for tuning EA parameters pays off in terms of EA performance: regardless of the tuning algorithm being used, the user is likely to obtain a much better EA than when relying simply on intuition and/or on the usual parameter setting conventions.

Czarn et al. [6] examined the statistical significance of the interaction between crossover and mutation and its relation to the increase in problem modality. They found that, as the test function's modality increases, the interaction between these two parameters becomes significant, and conjectured that for highly modal functions the possibility of interaction between them must be considered. The authors discuss the fact that in usual implementations of a GA the effect of the seed is not regulated and so the experimental design may be conceived as being entirely randomized, and they describe *blocking* as a solution for controlling the effect of different seeds. In this paper, this is accomplished by the replications used in the applied factorial design technique, which also studies the effect of all possible interactions amongst parameters.

III. Experimental Designs

The main goal of an experimental design is to obtain the maximum information about factors and their influence on the output of a system with the minimum number of experiments. A proper experiment analysis helps separating the effects of various factors that might affect the performance of the algorithm. It also allows determining whether a factor's effect is significant or the observed difference is simply due to random variations caused by measurement errors and uncontrolled parameters [8].

Consider that the *response variable* is the outcome of an experiment, generally represented by a performance metric or a response of the system. Consider also that *factors* are the variables that affect the response variable and have several alternative values called *levels*. Knowing that at the beginning of a performance study the number of factors and levels to be considered are usually large, a full factorial design with a large number of factors and levels may not be the best use of available resources. Therefore, it is important to reduce the number of factors and consider only those that have significant impact on the system's performance.

In this context, 2^k experimental projects can be used to determine the effect of k factors considering two values (i.e., levels) for each factor. Such values are usually taken as the minimum and maximum levels of a factor, based on the assumption that its effect is often unidirectional, i.e., the performance of the algorithm continuously increases

(or decreases) as the factor's value grows from minimum to maximum. The 2^k factorial design helps us decide if the difference in performance of two levels of a factor is significant enough to justify its further examination [8]. Apart from giving researchers a way to map the effect of different parameters in different problems, experimental designs may be used to optimize the response of a particular problem. Besides analyzing the factors individually, it also considers interactions between factors. Note that two factors A and B are said to interact if the effect of one depends on the level of the other.

Regarding a Genetic Programming algorithm, the response variable of the system is given by the fitness of the best individual in the end of the evolution, and parameters such as population size and maximum number of generations are taken as factors. Factor values are those values assumed by the GP parameters, such as a population size of 200 or 1,000.

A. Factorial Designs with Replication

As explained, 2^k factorial designs are useful for measuring the effects of parameters in experiments. However, considering that Genetic Programming is a stochastic method, experimental errors cannot be estimated with a single execution of the algorithm. Hence, in this case, $2^k r$ factorial designs are more appropriate: they follow the same principals of the 2^k design, but each parameter configuration is repeated r times.

A $2^k r$ factorial design works as follows (see [8] for more details). For each factor i a variable x_i is defined according to Eq. 1.

$$x_i = \begin{cases} -1, & \text{if } i \text{ is at the lower level} \\ 1, & \text{if } i \text{ is at the upper level} \end{cases} \quad (1)$$

The idea behind the factorial design is that the value of the response variable y can be regressed on x_i using a nonlinear model such as the one presented in Eq. 2, which considers a $2^2 r$ factorial design. In Eq. 2, q_0 is the mean value of the response variable, q_A is the effect of factor A on the response value, q_B is the effect of factor B, q_{AB} is the effect of the interaction between factors A and B, and e is the error.

$$y = q_0 + q_A x_A + q_B x_B + q_{AB} x_A x_B + e \quad (2)$$

A $2^k r$ factorial design is performed in four steps: (i) computing the effects, which are the coefficients ($q_i's$) in Eq. 2; (ii) allocating the variation explained by each factor; (iii) determining the confidence intervals of the effects to see which are significant[1]; (iv) verifying if the technique's assumptions are not violated.

1) Computing Parameters Effects: The coefficients of Eq. 2 are the effects of each factor or factor interaction ($q_i's$). The simplest way to compute these effects is to prepare a sign matrix. Table I shows the sign matrix for a $2^2 3$ factorial design (2 factors of 2 levels each, with 3 repetitions). The first column of the matrix (I) always consists of 1's. The next two columns (A and B) contain all possible combinations of -1 and 1. Column AB is the product of the entries in columns A and B. The twelve observations (three repetitions for each of the four parameter configurations) are listed in column y, whereas the average of the responses for each configuration is listed in column $\overline{y}$.

TABLE I
EXAMPLE OF SIGN MATRIX FOR A $2^2 3$ FACTORIAL DESIGN [8]

I	A	B	AB	y	Mean ($\overline{y}$)
1	-1	-1	1	(15, 18, 12)	15
1	-1	1	-1	(25, 28, 19)	24
1	1	-1	-1	(45, 48, 51)	48
1	1	1	1	(75, 75, 81)	77
164	86	38	20		Total
41	21.5	9.5	5		Total/2^2

Next, the sum of the products of the entries in column I and the mean responses in column $\overline{y}$ is put under column I (i.e., $1 \times 15 + 1 \times 24 + 1 \times 48 + 1 \times 77 = 164$) . The same is done for columns A, B, and AB. Following, the sums under each column are divided by 2^k (2^2 in the example) to give the effects of each factor, which correspond to the coefficients of the regression model.

Once the effects have been computed, the model can be used to estimate the response variable for any parameter configuration. The model for a $2^2 r$ design is defined by Eq. 3, where $\widehat{y}_i$ is the estimated response when factors A and B are at levels x_{Ai} and x_{Bi}, respectively. The experimental error is the difference between the estimate and the measured value in the jth replication of the ith experiment (y_{ij}). Note that the sum of all errors must be zero, and the sum of the squared errors (SSE) can be used to estimate the variance of the errors and also to compute confidence intervals for the effects.

$$\widehat{y}_i = q_0 + q_A x_{Ai} + q_B x_{Bi} + q_{AB} x_{Ai} x_{Bi} \quad (3)$$

2) Allocating Fitness Variation: A factor's importance is measured by the proportion of the total variation in the response that it explains. Thus, if two factors explain 90% and 5% of the response variation, respectively, the second may be considered unimportant when compared to the first. In a $2^k r$ design, the variation due to experimental errors can be isolated as well as the variations due to single factors and factor interactions. The total variation or total sum of squares (SST) is given by Eq. 4, where $\overline{y}_{..}$ denotes the grand mean, i.e., the mean of the responses from all replications of all experiments.

$$SST = \sum_{i,j} (y_{ij} - \overline{y}_{..})^2 \quad (4)$$

SST can be divided into parts: one for each factor or factor interaction and one for the error. In a $2^2 r$ design, where $k = 2$, SST is calculated by Eq. 5, which can be rewritten as Eq. 6, where SSA, SSB, and SSAB – the sum of squares due to each factor and factor interaction – are the variations explained by

[1]Consider the 90% confidence interval of factor A as being (a, b); this means that, in 90% of the time, the value of A is between a and b, inclusively. If the confidence interval of an effect includes zero, then that effect is not significant at that level of confidence.

factors A, B, and interaction AB, respectively, and SSE is the unexplained variation attributed to experimental errors.

$$SST = 2^2 r q_A^2 + 2^2 r q_B^2 + 2^2 r q_{AB}^2 + \sum_{i,j} e_{i,j}^2 \quad (5)$$

$$SST = SSA + SSB + SSAB + SSE \quad (6)$$

The sum of squares of the mean (SS0) is given by $SS0 = 2^2 r q_0^2$. Since $SSY = \sum_{i,j} y_{i,j}^2$, we have Eq. 7 and Eq. 8 for a $2^2 r$ design.

$$SSY = SS0 + SSA + SSB + SSAB + SSE \quad (7)$$

$$SST = SSY - SS0 \quad (8)$$

Dividing SSA by SST, we obtain the percentage of the variation explained by factor A. Similarly, dividing SSB, SSAB, and SSE by SST yields, respectively, the percentage of variation explained by factor B, by the interaction between factors A and B, and the percentage of unexplained variation attributed to experimental error.

3) Effects Confidence Intervals: According to Jain [8], if the variance of the sample estimates is known, the confidence intervals for the effects can be computed. Assuming that errors are normally distributed with zero mean and σ_e^2 variance, then it follows from the model that the y_i's are also normally distributed with the same variance. The variance of errors can be estimated from the SSE with Eq. 9, which is called the Mean Square of Errors (MSE). $2^k(r-1)$ is the number of independent terms in the SSE, since the r error terms of an experiment add up to zero. The variance of the mean and of the effects can be computed by Eq. 10.

$$s_e^2 = \frac{SSE}{2^k(r-1)} \quad (9)$$

$$s_{q_0} = s_{q_A} = s_{q_B} = s_{q_{AB}} = \frac{s_e}{\sqrt{2^k r}} \quad (10)$$

The confidence interval for each effect is calculated by Eq. 11. Any effect whose confidence interval does not include zero is significant at the given confidence level. The normal distribution (z) is used when there are thirty or more observations.

$$q_i \mp t_{[1-\frac{\alpha}{2};2^k(r-1)]} s_{q_i} \quad (11)$$

4) Assumptions: Some assumptions were made for deriving the expressions defined in the previous section, and they need to be verified in order to validate the model obtained. Some common assumptions are:

1) The model errors are statistically independent;
2) The model errors are additive;
3) The errors are normally distributed;
4) The errors have a constant standard deviation σ_e;
5) The effects of factors are additive.

In many cases, the assumption that the effects are additive is not valid. In this case, we should try a multiplicative model and compare the results with those obtained using the additive model. An additive model assumes there is no strong dependence between two parameters and that changing the value of one does not necessarily affect the other (parameter independence). The multiplicative model, in contrast, takes these cases into account.

If the effects are multiplicative, for instance, we must transform the measured responses to their logarithm and then use an additive model, keeping in mind that the "new" response variable $y'_{i,j}$ is in fact the logarithm of the original response variable $y_{i,j}$. After the analysis, we can take the antilog of the additive effects (q_i) to produce multiplicative effects ($u_i = 10^{q_i}$).

Some ways of verifying if the additive model does not represent the data are:

- The range of values covered by the response variable y is large, which means that taking the arithmetic mean is inappropriate and a log transformation is needed;
- A plot of residuals versus predicted response shows the residuals at the same order of magnitude of the response variable, i.e., the residuals are not small compared to the response. Also, the spread of the residuals may be large at larger values of the response. These indicate the need for a logarithmic transformation;
- The normal quantile-quantile plot of the residuals shows that the larger positive and negative residuals do not follow the same line as those in the middle. This indicates that the residuals have a distribution that has a longer tail than the normal distribution, i.e., the normality assumption is violated.

B. One-Factor Experiments

After defining the parameters with the largest effects on fitness, one-factor designs can be used to compare several alternatives of a single parameter while the others remain fixed. Any number of levels can be used, unlike with the 2^k designs, where only two levels can be tested.

The model used in this case is $y_{ij} = \mu + \alpha_j + e_{ij}$, where y_{ij} is the ith observation with the parameter at level j, μ is the mean response, α_j is the effect of level j and e_{ij} is the error term. The effects add up to zero [8].

1) Computing Parameters Effects: The data consists of r observations for each of the a alternative levels of the factor. The ar observations are arranged in a $r \times a$ matrix so that each column contains the r observations of the jth alternative (α_j), whose effect can be calculated by Eq. 12, where $\bar{y}_{.j}$ is the mean of the observations in column j, i.e., the average fitness of all repetitions of the same parameter level, and μ is the grand mean, i.e., the average of all ar observations.

$$\alpha_j = \bar{y}_{.j} - \mu \quad (12)$$

The estimated response for each of the a alternatives is $\hat{y}_j = \mu + \alpha_j$. The difference between the measured and the estimated response represents experimental error.

2) Allocating Experiments Variation: The total variation of the response variable (fitness) in a one-factor experimental design can be allocated to the studied parameter and to errors. Similarly to the $2^k r$ factorial design, we have Eq. 13, which is the same as Eq. 14.

$$SSY = SS0 + SSA + SSE \quad (13)$$

which is the same as

$$\sum_{i,j} y_{ij}^2 = ar\mu^2 + r\sum_j \alpha_j^2 + \sum_{i,j} e_{ij}^2 \quad (14)$$

SSE can be obtained without calculating individual errors. The total variation of y is defined by Eq. 15.

$$SST = SSY - SS0 = SSA + SSE. \quad (15)$$

SSA and SSE represent, respectively, the explained and unexplained parts of the variation. Dividing SSA by SST gives the percentage of the total variation explained by the parameter.

3) Analyzing Variance: A high percentage of explained variation indicates a good model. However, it is necessary to verify if the factor has statistically *significant* effect on the response. The F-test [8] is used to check whether SSA is significantly greater than SSE. If the value of Eq. 16 is larger than quantile $F_{[1-\alpha;a-1;a(r-1)]}$ obtained from the table of quantiles of F-variates, SSA is considered significantly higher than SSE, i.e. the factor is assumed to explain a significant fraction of the variation in the response variable.

$$\frac{SSA \times a(r-1)}{(a-1) \times SSE} \quad (16)$$

4) Effect Confidence Intervals: The confidence intervals for the effects are calculated similarly to the described in Section III-A.3. The standard deviations of errors, grand mean (μ) and effects (α_j) are calculated, respectively, by

$$s_e = \sqrt{\frac{SSE}{a(r-1)}} \qquad s_\mu = \frac{s_e}{\sqrt{ar}} \qquad s_{\alpha_j} = s_e\sqrt{\frac{a-1}{ar}}$$

The corresponding confidence intervals are calculated using Eq. 17 and Eq. 18. A confidence interval containing zero implies that the effect is not significant at the chosen level of confidence. For thirty or more observations, the normal distribution (z) is used instead of t.

$$(\mu \mp s_\mu t_{[1-\frac{\alpha}{2};r(a-1)]}) \quad (17)$$

$$(\alpha_j \mp s_{\alpha_j} t_{[1-\frac{\alpha}{2};r(a-1)]}) \quad (18)$$

IV. Computational Results

This section describes the experimental setup for running $2^k r$ factorial designs for two classical problems in the GP literature: the symbolic regression of function $x^4+x^3+x^2+x$ and the boolean 11-multiplexer problem [3]. These problems were chosen because they have already been exhaustively studied, so that near optimal configurations for the parameters are known.

Recall that the 2^k factorial design helps us to decide if the difference in performance of two levels of a factor is significant enough to justify further examination of that factor. Six factors were analyzed for both problems, namely population size, number of generations, maximum tree depth, and crossover, mutation and reproduction rates. Furthermore, a seventh parameter was considered for the 11-multiplexer problem: the maximum number of tree nodes. Therefore, a $2^6 r$ factorial design was performed for the symbolic regression and a $2^7 r$ for the 11-multiplexer. The value of r was set to thirty, a standard value in the literature.

TABLE II
Two Parameter Configurations of the Factorial Designs for Symbolic Regression, with lower (LB) and upper bound (UB) values.

Parameter	Code	Config. 1		Config. 2	
		LB	UB	LB	UB
Population Size	A	20	10000	350	650
Generations	B	5	250	35	65
Tree Depth	C	2	20	2	20
Crossover Rate	D	0.6	0.99	0.6	0.99
Mutation Rate	E	0	0.4	0	0.4
Reproduction Rate	F	0	0.2	0	0.2

The experiments were performed using the lil-GP system [10]. For both problems, the fitness of the best individual of the run was considered the response variable (y). One of the most difficult tasks when building a 2^k factorial design is to decide which lower and upper bound parameter values to use. These bounds greatly influence the results, considering that the results show whether or not increasing the parameter value from the lower to the upper bound significantly changes the performance of the algorithm. Experiments with two different configurations of lower and upper bounds were run for each problem.

As explained in Section III, there are some assumptions made when running a factorial design. One of them is that the effects of the factors is additive. Our experiments showed that this is not the case for GP algorithms. Hence, the parameter effects were calculated using a multiplicative model, and the fitness used (y') was $y' = \log(1+y)$ for symbolic regression and $y' = \log(y)$ for 11-multiplexer, where y is the "real" fitness.

This section primarily shows the effect of each of the studied parameters and their interactions. In a second phase, we focus on the parameters with the greatest impacts on fitness and fine tune them using a one factor design for the symbolic regression problem and common multiple linear regression techniques for the 11-multiplexer, according to the number of parameters found to be relevant by the factorial designs.

A. Symbolic Regression Problem

Table II shows the parameter values used in the factorial design for the symbolic regression in both configurations of lower bound (LB) and upper bound (UB) values. The choice of the lower and upper bounds of the first configuration considered really extreme values, specially for population size and number of generations. The second configuration redefined the bounds for population size and number of generations as $\pm 30\%$ of the values used by Koza [3].

The results for the $2^6 30$ factorial design are shown in

TABLE III
RESULTS OF THE $2^6 30$ FACTORIAL DESIGN FOR SYMBOLIC REGRESSION. SIGNIFICANT EFFECTS AND CORRESPONDING PERCENTAGE OF EXPLAINED VARIATION ARE PRESENTED.

Factor	Config. 1		Config. 2	
	%	Effect	%	Effect
A	**35.8**	-0.35	0.53	-0.03
B	**15.11**	-0.23	2.28	-0.07
C	**16.38**	-0.24	**69.00**	-0.37
E	0.63	-0.05	0.42	0.03
F	-	-	0.19	-0.02
AB	0.45	-0.04	-	-
AC	3.88	-0.12	0.15	-0.02
AE	0.61	0.05	-	-
BC	**10.37**	-0.19	1.91	-0.06
BE	0.56	-0.04	-	-
CE	-	-	0.95	0.04
CF	-	-	0.22	-0.02
ABC	1.47	-0.07	-	-
ABE	0.53	0.04	-	-
ABCE	0.05	0.01	-	-
% Explained	**86.07%**		**75.95%**	
Std Deviation	**0.005**		**0.005**	

Table III, which reports the effect[2] that each factor or factor interaction has on the response variable (fitness) and the percentage of the fitness variation that it explains. Note that, for the sake of simplicity, only significant results (representing those effects whose 99% confidence interval does not contain zero) are shown[3]. The factors or interactions which explain the largest variation percentages are in bold. The last but one line in Table III shows the percentage of fitness variation explained by all factors and factor interactions combined. These figures show us that, with 30 repetitions, 86.07% of the experimental variation is explained by the parameters, while the remaining 13.93% are attributed to experimental error (which may be random effects such as the value of the seed used to generate random numbers during the experiment [6]). The last line presents the standard deviation of the effects, which can be used to calculate their corresponding confidence intervals.

The results show that the parameters with the largest influence on the GP's performance for the symbolic regression problem are A, B and C (i.e., population size, number of generations and maximum tree depth, as defined in Table II) for the first configuration, and only C (i.e., tree depth) for the second configuration, where tree depth alone explains 69% of the fitness variation. The influence of A and B in the first but not in the second parameter configuration is closely related to the lower and upper bounds chosen for these parameters: since the total number of fitness evaluations varies in orders of magnitude from the lower to the upper bound in the first configuration, the impact of these parameters will certainly be large. In the second configuration, when the number of evaluations performed considering the lower and upper bounds is less discrepant, only the effect of tree depth appears as being significant. Also note that the effects of changing the rates of the genetic operators from lower to upper bounds are small: using a 60% crossover rate with no mutation or reproduction yields the same results as using 99% of crossover and 40% of mutation. Lastly, the results show that, despite the low percentage of fitness variation they explain, the effect of various parameter interactions is statistically significant, which further suggests the inadequacy of the fractional factorial designs for GP algorithms.

Based on the results obtained by the second configuration, we fixed the values of all parameters but the tree depth in the center of the lower and upper bounds, so that we used 500 individuals, evolving over 50 generations, with 90% crossover, 10% reproduction and no mutation. Then, we performed a one-factor design in order to pinpoint the tree depth that yields the best results, varying the tree depth in a pace of 2 from 2 to 20 (i.e., 2, 4, ..., 20). Table IV presents the results of the one-factor design, showing the average fitness for each tree depth ($\bar{y}_{.j}$), the effect of that level (α_j) and the 99% confidence intervals for the effects. Again, for the sake of simplicity, only the statistically significant effects (those whose 99% confidence intervals do not contain zero) are shown. Recall that, if the confidence interval contains zero, that means that the corresponding effect may be null and, therefore, it is not significant.

For the symbolic regression problem, the fitness is defined as the sum of errors of the individual on the fitness cases. Therefore, the lower the fitness value, the better the individual is. With this in mind, we can observe in Table IV that the best results are obtained with tree depth eight, since α_8 is the largest negative effect (if the tree depth is eight, according to the obtained model, 4.01 will be subtracted from the fitness).

We conclude that, for the symbolic regression problem, the parameter configuration which produces the best results is to evolve 500 individuals over 50 generations, using a maximum tree depth of 8, 90% and 10% crossover and reproductions rates respectively, and no mutation. With this configuration, the average fitness of 30 repetitions was 1.83. To reach this conclusion, a total of 4140 experiments were run, which comprise 30 repetitions of 64 (2^6) parameter configurations for each factorial design and 30 repetitions of 10 parameter configurations for the one-factor design.

B. 11-Multiplexer Problem

Table V shows the two parameter value configurations used in the factorial design for the 11-multiplexer. Note that for this problem we performed a $2^7 30$ factorial design, as the maximum number of tree nodes was also taken into account. Again, the choice of bounds for the first configuration considered really extreme values, specially for population size and number of generations and the second configuration redefined the bounds for population size and number of generations as $\pm 30\%$ of the values used by Koza [3] and upper bound of the

[2]The effect of a factor or factor interaction is its coefficient in the logarithmized model $y' = q_0 + q_a x_a + q_b x_b + \ldots + q_{abcdef} x_{abcdef}$

[3]Tables with the complete results can be downloaded from `http://www.dcc.ufmg.br/~eblima/CEC2010.tar.bz2`

TABLE IV

ONE-FACTOR DESIGN RESULTS FOR SYMBOLIC REGRESSION. RESULTS PER TREE DEPTH: AVERAGE FITNESS, EFFECT AND 99% CONFIDENCE INTERVAL OF EACH EFFECT. $\mu = 5.84$. THE TREE DEPTH EXPLAINS 63.43% OF THE TOTAL FITNESS VARIATION.

	H2	**H6**	**H8**	**H10**	**H12**	**H14**	**H16**
$\bar{y}_{.j}$	30.50	2.78	1.83	1,95	2.06	2.29	2.55
α_j	24.67	-3.06	-4.01	-3.89	-3.77	-3.55	-3.28
CI	(21.79, 27.53)	(-5.93, -0.19)	(-6.88, -1.14)	(-6.76, -1.02)	(-6.64, -0.91)	(-6.42, -0.68)	(-6.15, -0.42)

TABLE V

TWO PARAMETER CONFIGURATIONS OF THE FACTORIAL DESIGNS FOR 11-MULTIPLEXER, WITH LOWER (LB) AND UPPER BOUND (UB) VALUES.

Parameter	**Code**	**Config. 1**		**Config. 2**	
		LB	**UB**	**LB**	**UB**
Population Size	A	20	1000	2800	5200
Generations	B	5	250	35	65
Tree Depth	C	2	20	2	20
Max. Tree Nodes	D	500	∞	500	1000
Crossover Rate	E	0.6	0.99	0.6	0.99
Mutation Rate	F	0	0.4	0	0.4
Reproduction Rate	G	0	0.2	0	0.2

TABLE VI

RESULTS OF THE $2^7 30$ FACTORIAL DESIGN FOR 11-MULTIPLEXER. SIGNIFICANT EFFECTS AND CORRESPONDING PERCENTAGE OF EXPLAINED VARIATION ARE PRESENTED.

Factor	**Config. 1**		**Config. 2**	
	%	**Effect**	**%**	**Effect**
A	**38.52**	0.04	1.36	3E-3
B	**29.23**	0.03	**18.28**	0.01
C	**5.16**	0.01	**42.18**	0.016
D	-	-	0.12	-8E-4
F	0.78	6E-3	0.29	-1E-3
G	-	- 0.05	6E-4	
AB	**7.15**	0.02		
AC	1.26	7E-3	1.36	3E-3
AD	-	-	0.05	-5E-4
AF	0.99	-6E-3	-	-
BC	**5.99**	0.02	**18.28**	0.01
BF	0.81	6E-3	-	-
BG	0.02	1E-3	-	-
CD	-	-	0.12	-8E-4
CF	0.04	1E-3	0.29	-1E-3
CG	0.02	8E-4	0.05	6E-4
ABC	1.76	8E-3	-	-
ABF	0.59	-5E-3	-	-
ACD	-	-	0.05	-5E-4
ACF	0.04	-1E-3	-	-
ABCF	0.07	-2E-3	-	-
% Explained	**92.57%**		**82.94%**	
Std Deviation	**0.0003**		**0.0002**	

maximum number of nodes as 1,000 to make the experiments feasible considering the available computational resources.

The results of the factorial designs are presented in Table VI, which has the same format as Table III and reports the significant effects[4] of factors or factor interactions and the percentage of fitness variation it explains. Note that an effect is considered significant if its 99% confidence interval does not include zero (a confidence interval containing zero means that the effect might be null, i.e., it is not significant)[5]. The factors or factor interactions which explain the largest variation percentages are in bold. The last but one line shows the percentage of variation jointly explained by all factors and factor interactions. The last line shows the standar deviation of the effects, which is used to calculate their corresponding confidence intervals.

We can observe in Table VI that, as with the symbolic regression problem, there is a large influence of factors A, B and C (i.e., population size, number of generations and tree depth) for the first configuration, explaining 38.52, 29.23 and 5.16% of the fitness variation respectively. For the second configuration, there is a large influence of factors B and C, which respectively account for 18.28 and 42.28% of the variation. It is interesting to note that the interaction between factors B and C explains 18% of the variation in the second configuration.

For the 11-multiplexer, since two factors were considered the most important ones in the second configuration, we fixed the values of the other five parameters and fine-tuned the number of generations and tree depth using multiple linear regression. We used 1,000 individuals, 60% crossover, zero mutation and reproduction and did not impose a limit on the number of nodes in a tree. We performed 30 repetitions of all combinations of 35, 40, 45, 50, 55, 60 and 65 generations and maximum tree depth of 5, 7, 9, 10, 11, 13, 15. Using the average of the 30 repetitions for each parameter configuration, we were able to adjust the model presented in Eq. 19, where y is the fitness, x_B represents the number of generations and x_C, the tree depth.

$$y = 0.69849 + 0.00195x_B - 0.00187x_C \qquad (19)$$

This model has $R^2 = 0.89$, which means factors B and C explain 89% of the variation in y. The 99% confidence intervals of factors B and C are, respectively, (0.00174,

[4]The effect of a factor or factor interaction is its coefficient in the logarithmized model $y' = q_0 + q_a x_a + q_b x_b + \ldots + q_{abcdefg} x_{abcdefg}$

[5]Tables with the complete results can be downloaded from `http://www.dcc.ufmg.br/~eblima/CEC2010.tar.bz2`

0.00216) and (-0.00253, -0.00121). Since neither interval contains zero, both parameters are statistically significant.

What the regression model in Eq. 19 tells us is that y increases in 0.00195 for each additional generation and decreases in 0.00187 for each additional level of tree depth. Since for the 11-multiplexer the higher the fitness, the better the individual is, we can conclude that a large number of generations and a small limit of tree depth produce the best results. In fact, in our experiments the best results were obtained with 65 generations and tree depth of 5, which were the largest number of generations and lowest tree depth we considered for this regression. We must point out, however, that considering that the 11-multiplexer consumes a great deal of computational resources, the tradeoff between better results and resource usage must be considered.

For the 11-multiplexer problem, a total of 5310 experiments were performed: 30 repetitions of 128 different parameter configurations for each factorial design and 30 repetitions of 49 different configurations for the multiple linear regression.

C. General Conclusions

The 2^k30 factorial designs performed in two classical genetic programming problems showed us that the tree depth has a high influence in the results, followed by the population size and number of generations. It is clear to us that the lower and upper bounds chosen also have an impact in the results of these experiments, and here they were initially chosen using an ad hoc approach. The values for mutation, crossover, reproduction and tree depth were set according to the most common values used in the literature. The most difficult values to choose were the population size and the number of generations.

Moreover, the information provided by the 2^k factorial design allows us to focus computational resources on the parameters that are more likely to influence the results of the GP. Here, two different approaches were used, depending on the number of parameters considered most relevant.

V. Conclusions and Future Work

One of the priorities of the Evolutionary Algorithms community is to improve the process of parameter setting, which has two main approaches: parameter tuning and parameter control. In this work we described a new methodology for parameter tuning og Genetic Programming algorithms. Our proposed methodology involves the application of Experimental Design techniques, namely factorial designs, one-factor designs and multiple linear regression.

The experiments show that factorial designs can be used to determine which parameters impact the algorithm's performance the most. With this information at hand, we focused parameter setting efforts on a much narrower portion of the parameter search space, enhancing the process as a whole.

We developed and tested our methodology using two classical problems in the GP literature: the symbolic regression of function $x^4 + x^3 + x^2 + x$ and the boolean 11-multiplexer problem. For both problems we considered six parameters, namely population size, number of generations, maximum tree depth, and crossover, mutation and reproduction rates. For the 11-multiplexer, we also studied the maximum number of nodes in a tree. For each parameter configuration we run 30 repetitions. Thus, we performed a $2^6 30$ factorial design for the symbolic regression problem and a $2^7 30$ factorial design for 11-multiplexer.

The results of the factorial designs show that the maximum tree depth is the parameter with the largest effect on the algorithm's performance, followed by the number of generations and the population size. After determining the most influential parameters, we used a one-factor design to fine-tune the tree depth for the symbolic regression problem and were able to conclude that the maximum tree depth of eight yields the best results for this problem. For the 11-multiplexer problem, the factorial design indicated tree depth and number of generations as most important parameters, so we fixed the other parameters and regressed these two using multiple linear regression. The model we obtained tells us that the larger the number of generations and the smaller the tree depth, the better. However, there is a tradeoff between better results and resource usage, considering that the 11-multiplexer is costly and excessively increasing the number of generations may be extremely time consuming depending on the other parameter values.

As future works, we intend to study better ways for setting the lower and upper bounds of the 2^k factorial project, as well as to test the methodology on GPs used in very different real world application domains. Furthermore, we believe this type of experimental work can help the study of other GP issues, such as the impacts of crossover versus mutation, etc.

References

[1] A. Eiben, Z. Michalewicz, M. Schoenauer, and S. J.E., *Parameter Control in Evolutionary Algorithms*, ser. Studies in Computational Intelligence (SCI). Springer, 2007, vol. 54.

[2] S. Smit and A. Eiben, "Comparing parameter tuning methods for evolutionary algorithms," in *IEEE Congress on Evolutionary Computation (CEC '09)*, 2009.

[3] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, 1992.

[4] Volker, S. K. Smit, and A. E. Eiben, "Costs and benefits of tuning parameters of evolutionary algorithms," in *Parallel Problem Solving from Nature PPSN X*, 2008, pp. 528–538.

[5] W. Banzhaf, P. Nordin, R. Keller, and F. Francone, *Genetic Programming – An Introduction. On the automatic evolution of computer programs and its applications*. Germany: Morgan Kaufmann, 1998.

[6] A. Czarn, C. MacNish, K. Vijayan, and B. Turlach, "Statistical exploratory analysis of genetic algorithms: the importance of interaction," in *Evolutionary Computation, 2004. CEC2004. Congress on*, vol. 2, June 2004, pp. 2288–2295 Vol.2.

[7] V. Nannen and A. Eiben, "Efficient relevance estimation and value calibration of evolutionary algorithm parameters," in *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*, Sept. 2007, pp. 103–110.

[8] R. K. Jain, *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation and Modeling*. New York, NY: Wiley - Interscience, April 1991.

[9] R. Feldt and P. Nordin, "Using factorial experiments to evaluate the effect of genetic programming parameters," in *European Conference on Genetic Programming*, vol. 1802. Springer-Verlag, 2000.

[10] D. Zongker and D. B. Punch, *lil-gp 1.01 User's Manual*, Michigan State University, 1996.