# A similarity measure for Straight Line Programs and its application to control diversity in Genetic Programming

R. Rueda[a,*], M.P. Cuéllar[a], L.G.B. Ruiz[a], M.C. Pegalajar[a]

[a]*Department of Computer Science and Artificial Intelligence, University of Granada, C/. Pdta. Daniel Saucedo Aranda s.n., 18071 Granada, Spain*

**Abstract**

Finding a balance between diversity and convergence plays an important role in evolutionary algorithms to avoid premature convergence and to perform a better exploration of the search space. In the case of Genetic Programming, and more specifically for symbolic regression problems, different mechanisms have been devised to control diversity, ranging from novel crossover and/or mutation procedures to the design of distance measures that help genetic operators to increase diversity in the population. In this paper, we start from previous works where Straight Line Programs are used as an alternative representation to expression trees for symbolic regression, and develop a similarity measure based on edit distance in order to determine how different the Straight Line Programs in the population are. This measure is used in combination with the CHC algorithm strategy to control diversity in the population, and therefore to avoid local optima to solve symbolic regression problems. The proposal is first validated in a controlled scenario of benchmark datasets and it is compared with previous approaches to promote diversity in Genetic Programming. After that, the approach is also evaluated in a real world dataset of energy consumption data from a set of buildings of the University of Granada.

*Keywords:* Diversity, Edit distance, Symbolic Regression, Genetic Programming, Straight Line Program

*Corresponding author

*Email addresses:* `ramonrd@ugr.es` (R. Rueda ), `manupc@decsai.ugr.es` (M.P. Cuéllar), `bacaruiz@ugr.es` (L.G.B. Ruiz), `mcarmen@decsai.ugr.es` (M.C. Pegalajar)

## 1. Introduction

In evolutionary computation, diversity and convergence play an important role in the exploration of the search space. As many authors argued (Burke et al., 2004; Badran & Rockett, 2007; Chen et al., 2009), finding a balance between diversity and convergence is critical in genetic algorithms, since premature convergence causes the end of the evolution in local optima, while an uncontrolled divergence may reduce exploitation of the search space. Two main mechanisms that guide the exploration of the search space in genetic algorithms have been identified in the literature (Smit & Eiben, 2010): *Variation*, which promotes diversity, and *Selection* to reinforce convergence. A suitable combination of these mechanisms helps to explore the search space to avoid falling in local optima (Črepinšek et al., 2013). Indeed, many approaches emerged to tackle the problem of premature convergence in genetic algorithms by proposing new evolutionary algorithms or improving genetic operators with the aim of delaying a premature convergence. For example, the work (Mc Ginley et al., 2011) implemented *ACROMUSE*, a genetic algorithm that adapts crossover, mutation and parameter selection to preserve diversity in the population. Lozano et al. (Lozano et al., 2008) proposed replacement strategies that consider both fitness quality and the diversity of an individual in the population, in order to maintain individuals with high fitness and diversity for the next generation. Aslam et al. (Aslam et al., 2018) presented a selection operator that determines whether two individuals can be recombined considering their distance. On the other hand, other authors established a criterion that helps to select the individuals that will be combined: e.g. techniques based on neighborhoods such as niching methods (Martín et al., 2016) or approaches that consider behavior similarities by using fitness sharing (Ekárt & Németh, 2000, 2002). A recent article showed how multi-objective optimization can be used to promote diversity in the population, considering both fitness and a diversity measure as objectives to be optimized (Segura et al., 2017).

In addition to the aforementioned problems of diversity and convergence, Genetic Programming (GP) has to deal with additional issues regarding the solution encoding (Koza, 1992). As the encodings used in GP have a non-linear structure, such as trees, it is harder to tackle the control of diversity (Burke et al., 2004). The problem of tree uncontrolled growth, known as the *bloating problem* in GP, leads to premature convergence (dal Piccol Sotto & de Melo, 2016). Preventing this problem is an implicit goal for researchers in GP, and different authors have proposed to modify genetic operators, fitness evaluation or selection schemes (Alfaro-Cid et al., 2008; Liu et al., 2007; de Jong et al., 2001) to solve the bloating problem while maintaining diversity in the population. Diversity measures may be classified into three main categories: *(a) behavioural or phenotypic diversity*, that considers differences in solution performance (fitness value) (Kalkreuth et al., 2015; Hildebrandt & Branke, 2015; Li et al., 2016), *(b) syntactic or genotypic diversity*, which computes structural differences between individuals (shape and content of solutions in the population) (Qu et al., 2015; Ferdjoukh et al., 2017) and *(c)* a combination of both previous approaches (Affenzeller et al., 2017; Kelly et al., 2019).

In this piece of research, we focus on improving diversity in formal grammar evolution by studying a combination of both phenotypic and genotypic approaches. As the solutions in GP are encoded using tree data structures traditionally, genotypic diversity measures focus on this type of representation (Ekárt & Németh, 2000, 2002; Burks & Punch, 2017; Pawlik & Augsten, 2016; Kulunchakov & Strijov, 2017; Burlacu et al., 2019). We may classify the cited methods as distance measures or metrics: whereas a metric holds the properties of non-negativity, identity, symmetry, and triangle inequality, the remaining distance measures fail to accomplish one or more of these properties (usually the triangle inequality), but they can provide a value to estimate how distant two encoded solutions are, and have provided good results in the problems they have been used. Examples of (non-metric) distance measures are described in Burks et al. (Burks & Punch, 2017), which implement a density measure that considers a portion of each tree and determine how genetic material is distributed in the population; or the work (Burlacu et al., 2019), that uses isomorphic properties to measure structural diversity between two trees as the number of common nodes. Regarding metric proposals, Mateusz et al. (Pawlik & Augsten, 2016) developed a metric that determines the differences between two individuals as the sequence of minimum cost of operations needed to transform one tree into another. Besides, Ekárt and Neméth described in (Ekárt & Németh, 2000, 2002) a metric that computes the structural difference of two encoded programs, distinguishing terminal and operator nodes.

Regarding phenotypic or behavioural diversity in genetic programming, the literature offers a wide variety of works that obtain semantic information from individuals during the evolutionary process and it is used it to improve the search space exploration in GP. These works range from classical methods such as the traditional *Ramped Half and Half* method to prevent the insertion of duplication trees into the population (Koza, 1992) to more recent works such as (Castelli et al., 2015) that proposed the Geometric Semantic Genetic Programming (GGSP) (Moraglio et al., 2012) algorithm that designs an operator which measures semantic differences between two individuals to guide the search space exploration, or Nguyen et al. (Uy et al., 2010) whose developed a Semantic Similarity Crossover (SSC) which add semantic knowledge to control the changes of the semantic of individuals by comparing similarities of random subtrees. In summary, the main works proposed to directly or indirectly control diversity in Genetic Programming go from the structures used to represent the population to genetic operators and measures to control the population growth (Ursem, 2002).

In this piece of research, we focus on improving diversity in GP in two ways: (i) studying alternative structures to classical trees and (ii) developing measures to control diversity during the genetic procedure for these alternative structures. In previous works, we studied an alternative representation scheme to tree encoding, using Straight Line Programs (SLP) (Rueda et al., 2019), and we concluded that using this representation may help to overcome limitations of classic tree encoding and to overcome local optima solutions. In this article, our main objective is to develop a metric based on edit distance that allows us to quantify how different two SLPs are, and use this metric to measure diversity in a population of SLPs to find a balance between diversity and convergence that helps to improve the exploration of the search space. More specifically, we combine the developed metric distance with the CHC algorithm (Eshelman, 1991), to achieve a balance in the exploration and exploitation of the search space. Thus, the main novelty presented in this manuscript is the design of the similarity measure for Straight Line Programs, the proof that this measure is a metric, and its application in combination with a well tested evolutionary scheme such as CHC to prove its practical application. We remark that the classic edit distance is applied over sequences, and the similarity measure proposed in this work is adapted to grammars as formal languages. As no previous works have been proposed to quantify the distance between Straight Line Programs, we test our approach against tree-based encodings as baseline methods.

The remaining of the manuscript is structured as follows: Section 2 describes the background of our research introducing the fundamentals of Symbolic Regression, the representation problem and an outline of the classic CHC algorithm. Section 3 works out the proposed similarity measure. Section 4 applies the proposed metric in combination with the CHC algorithm to control diversity and convergence in genetic programming. Section 5 shows the experimental results in synthetic data and real energy consumption data and discusses the comparative study of the proposal with state-of-the-art algorithms. Finally, Section 6 summarizes the conclusions obtained and describes future works.

## 2. Background

### *2.1. Symbolic regression and the representation problem*

Regression analysis (Harrell, 2015) is a statistical method that allows to find the relationships between dependent and independent variables. More specifically, regression analysis is composed by a model hypothesis $f(\bar{x}, \bar{w}) + \epsilon$, a set of input data $\bar{x} = \{x_1, x_2, ..., x_n\}$, a set of output data $\bar{y} = \{y_1, y_2, ..., y_m\}$, a set of constant parameters $\bar{w} = \{w_1, w_2, ..., w_k\}$, and an error $\epsilon$ that represents the part of the data that the model $f(\bar{x}, \bar{w})$ is unable to model. The main goal of regression analysis is to approximate the best values for the parameters $\bar{w}$ such that $\bar{y} \approx f(\bar{x}, \bar{w})$. With the aim of estimating the parameters, an error function is minimized such as $e(f, \bar{y}) = ||\bar{y} - f(\bar{x}, \bar{w})||$ as the sum of squared errors between the estimated functional model $f(\bar{x}, \bar{w})$ and the model hypothesis response $\bar{y}$.

The main limitation of regression analysis arises when the model hypothesis $f$ is unknown and it is difficult to formulate manually. To solve this limitation, symbolic regression (SR) (Billard & Diday, 2002) combines a set of primitive operators (such as $+, -, *, /$), independent variables $\bar{x}$ and parameters $\bar{w}$ to build an algebraic expression $\tilde{f}$ as an approximation to the optimal model $f$. Since symbolic regression is a NP-hard problem (Lu et al., 2016), Genetic Programming (Koza, 1992) or Grammatical Evolution (O'Neill & Ryan, 2001) algorithms have been traditionally used to explore the search space and to find the best approximation $\tilde{f}$ that minimizes an error measure with respect to the desired output data. GP is an evolutionary method based on biological evolution and simulates the evolutionary process. More specifically, GP builds a population of individuals which stochastically transforms into a new population in order to simulate the evolutionary cycle. During that cycle, it is expected that the best individual survives and will contribute to a better population. Similarly, different mechanisms were studied to simulate the evolutionary cycle (crossover, mutation, etc) (Angeline, 1994) as well as the individual representation. Indeed, with regards to SR problems, tree structures have been highly used to encode algebraic expressions (McKay et al.,

1995) achieving promising results. As the representation problem determines the size of the search space, recent studies proved that alternative representations may reduce the search space, such as: linear genetic programming (Brameier & Banzhaf, 2001), that encodes programs as a sequence of instructions that operate over a memory equipped with a set of registers, instruction matrix (Li et al., 2008) that evolves tree nodes and subtrees separately, or linear strings of integers (Miller & Thomson, 2000) that encode a graph as a list of node connections and functions. Also, this is the case of Straight Line Programs (Alonso et al., 2008; Rueda et al., 2019).

## *2.2. Straight Line Programs*

A Straight Line Program (SLP) encodes a Straight Line Grammar (SLG) in Chomsky Normal Form (Claude & Navarro, 2008). A SLG is a context-free non-recursive grammar $(V, T, P, S)$ able to generate a language with a single word, where $V$ is the set of variables/non-terminal symbols of the grammar, $T$ is the set of terminal symbols, $P$ is the set of production rules and $S$ is the starting non-terminal symbol of the grammar. Then, a SLP encodes a set of SLG production rules that can be used in SR to generate a single algebraic expression. In the SR problem addressed in this work, the set of terminal symbols is $T \equiv O_u \cup O_b \cup X \cup W$, where $O_u$ is a set of unary operators, $O_b$ is a set of binary operators, $X$ is a set of terminal input data variables $\{x_1, x_2, ..., x_n\}$, and $W$ is a set of constant parameters $\{w_1, w_2, ..., w_k\}$. A SLP contains $N$ production rules $U_1, U_2, ..., U_N \in V$, where $U_N$ is the starting symbol of the grammar and each production rule is of the form $U_i \rightarrow o_u r_{i1}$ or $U_i \rightarrow o_b r_{i1} r_{i2}$, where $o_u \in O_u$, $o_b \in O_b$ are operators, and $r_{i_1}, r_{i2} \in X \cup W \cup \{U_{i-1}, U_{i-2}, ..., U_1\}$ are the first and second operands, which can be a data terminal symbol or a non-terminal symbol that references subsequent production rules to avoid recursion.

In SR, a subset of elements in $O_b$ can have the commutative property. In these cases, we decide to establish an order for the construction of each production rule to reduce the search space. If a rule uses a commutative operator, then $r_{i_1} \prec r_{i_2}$ must be fulfilled, where the partial order relationship $\prec$ is defined as: $x_i \prec x_j \iff i < j$, $w_i \prec w_j \iff i < j$, $U_i \prec U_j \iff i < j$, $x_i \prec w_j \forall i, j$, and $w_i \prec U_j \forall i, j$. Equation 1 shows a sample SLP (A) and its representation (A') considering the partial order relationship constraint. As it can be seen, such constraint reduces the search space without shrinking the space of possible solutions to a SR problem. While the SLP A generates the algebraic expression $A = (\frac{w_3}{w_2} - w_1) * x_1 + \frac{w_3}{w_2} - w_1$, the SLP A' derives the equivalent expression $A' = \frac{w_3}{w_2} - w_1 + x_1 * (\frac{w_3}{w_2} - w_1)$.

$$A = \begin{cases} U_0 \rightarrow / \; w_3 \; w_2 \\ U_1 \rightarrow - \; U_0 \; w_1 \\ U_2 \rightarrow * \; U_1 \; x_1 \\ U_3 \rightarrow + \; U_2 \; U_1 \end{cases} \Longleftrightarrow A' = \begin{cases} U_0 \rightarrow / \; w_3 \; w_2 \\ U_1 \rightarrow - \; U_0 \; w_1 \\ U_2 \rightarrow * \; x_1 \; U_1 \\ U_3 \rightarrow + \; U_1 \; U_2 \end{cases} \quad (1)$$

### *2.3. CHC Algorithm*

The CHC algorithm (Cross generational elitist selection, Heterogeneous recombination, and Cataclysmic mutation) is an evolutionary algorithm proposed by Eshelman (Eshelman, 1991) for binary encoding, and it was designed to hold a balance between diversity and convergence in the population. This algorithm was adapted for real-coded chromosomes in (Eshelman & Schaffer, 1993) and (Cordón et al., 2006). Unlike a classical genetic algorithm, CHC does not use a mutation operator, and introduces four components to achieve the aforementioned balance:

- An elitist selection. The $N$ best individuals of the current and generated offspring populations are selected to compose the new population in the next generation, where $N$ stands for the population size.
- The HUX (original binary encoding proposal (Eshelman, 1991)) or BLX-$\alpha$ (extended real-coded CHC (Eshelman & Schaffer, 1993)) crossover operators, to avoid premature convergence caused by recombination.
- An incest prevention mechanism to avoid crossover of similar solutions. If a distance measure over two parent solutions is over a threshold $\tau$, then the parent crossover is allowed. In the original binary CHC proposal, the Hamming distance was used to compare parent chromosomes, and the value $\tau$ was calculated initially as $L/4$ ($L$ is the size of the chromosomes). On the other hand, the real-coded CHC used the Euclidean distance to measure the similarity between two parents, and $\tau$ was initialized to $0.1 * d_{max}$ ($d_{max}$ is the maximum distance between two elements in the population).
- A restart procedure to reinitialize the population if it has converged. If two populations in consecutive algorithm iterations contain the same solutions, then $\tau$ is decreased. When

$\tau \leq 0$, the population is reinitialized with random solutions and a copy of the best solution found during the evolutionary process.

In this article, we use a variant of the real-coded CHC algorithm in Section 4 adapted to the SLP encoding scheme. We use the metric proposed in Section 3 as a diversity measure between SLPs.

## 3. Similarity measure for Straight Line Programs

Our goal is to define a similarity measure that provides the structural difference between SLPs and can be computed efficiently. We are inspired by the edit distance metric (Ristad & Yianilos, 1998). Hence, the proposed similarity measure provides the minimum number of operations required to transform one SLP into another. As in edit distance, the available operations to compute such transformation are insertions, deletions and substitutions. Then, the more similar two SLPs are, the lower the proposed distance value should be and in contrast, the more dissimilar two SLPs are, the greater value the measure should provide.

Given two SLPs $A$ and $B$ coming from two SLGs $G_A = (V, T, P_A, A_s)$ and $G_B = (V, T, P_B, B_s)$, we define the similarity measure between both SLPs as $d(A, B) = d(A_s, B_s)$, i.e. the similarity measure between the starting symbols of each SLP. The definition of $d(A_s, B_s)$ is provided as a recursive formula with general and base cases.

**Base case.** Let $t_1, t_2 \in T \cup \{\epsilon\}$ be two terminal symbols of the grammar (operators, data variables, constant parameters, or the empty word). Then we define $d(t_1, t_2)$ as shown in Equation 2.

$$d(t_1, t_2) = \begin{cases} 1, & t_1 \neq t_2 \\ 0, & t_1 = t_2 \end{cases} \tag{2}$$

In the base case, $d(t_1, t_2) = 1$ means a substitution of $t_1$ with $t_2$, $d(t_1, \epsilon)$ means deletion of $t_1$, and $d(\epsilon, t_2)$ means insertion of $t_2$.

**General case.** For the general case, we use the following notation: $o_i^A, o_j^B \in O_u \cup O_b$ are the operators used in the $i$-th and $j$-th rules of SLPs $A$ and $B$, respectively; $r_{i1}^A, r_{i2}^A, r_{j1}^B, r_{j2}^B \in V \cup T \cup \{\epsilon\}$ are the first and second operands of the $i$-th and $j$-th rules of SLPs $A$ and $B$. We focus on rules of the form $U_i^A \rightarrow o_i^A r_{i1}^A r_{i2}^A$ and $U_j^B \rightarrow o_j^B r_{j1}^B r_{j2}^B$ without loss of generality. In case $o_i^A$ (respectively

$o_j^B$) is an unary operator, then it is assumed that $r_{i2}^A = \epsilon$ (respectively $r_{j2}^B = \epsilon$). We also distinguish a set $C \in O_b$ as the subset of binary operators that meet the commutative property. With this in mind, Equation 3 describes how to compute the distance $d(U_i^A, U_j^B)$ between the aforementioned rules.

$$d(U_i^A, U_j^B) = \begin{cases} d(o_i^A, o_j^B) + d(r_{i1}^A, r_{j1}^B) + d(r_{i2}^A, r_{j2}^B) \text{ if } o_i^A, o_j^B \in \{O_u \cup O_b\} \backslash C \\ d(o_i^A, o_j^B) + min\{d(r_{i1}^A, r_{j1}^B) + d(r_{i2}^A, r_{j2}^B), \\ \quad d(r_{i1}^A, r_{j2}^B) + d(r_{i2}^A, r_{j1}^B)\} \text{if } o_i^A \in C \vee o_j^B \in C \end{cases} \tag{3}$$

In Equation 3, we may observe that the similarity measure between rules $U_i^A$ and $U_j^B$ computes the minimum number of insertion, deletion and substitution operations to transform $U_i^A$ into $U_j^B$, considering special cases when the commutative property allows to exchange the order of the rule operands.

A final consideration must be taken into account when computing the distance between operands, as for instance $d(r_{i1}^A, r_{j1}^B)$. We have defined the similarity measure for terminal symbols in Equation 2 and for non-terminal symbols in Equation 3. As two arbitrary rule operands (named as $u_1, u_2 \in V \cup T$) being compared can be terminal or non terminal indistinguishably, we define $d(u_1, u_2)$ for these cases as Equation 4 shows.

$$d(u_1, u_2) = \begin{cases} d(o_1, \epsilon) + d(r_{11}, u_2) + d(r_{12}, \epsilon) \text{ if } u_1 \in V, u_2 \in T, o_1 \in \{O_u \cup O_b\} \backslash C \\ d(o_1, \epsilon) + min\{d(r_{11}, u_2) + d(r_{12}, \epsilon), \\ \quad d(r_{11}, \epsilon) + d(r_{12}, u_2)\} \text{ if } u_1 \in V, u_2 \in T, o_1 \in C \\ d(\epsilon, o_2) + d(u_1, r_{21}) + d(\epsilon, r_{22}) \text{ if } u_1 \in T, u_2 \in V, o_2 \in \{O_u \cup O_b\} \backslash C \\ d(\epsilon, o_2) + min\{d(u_1, r_{21}) + d(\epsilon, r_{22}), \\ \quad d(\epsilon, r_{21}) + d(u_1, r_{22})\} \text{ if } u_1 \in T, u_2 \in V, o_2 \in C \end{cases} \tag{4}$$

The first two cases in Equation 4 assume that $u_1 \to o_1 r_{11} r_{12}$ is a non-terminal symbol and $u_2 \in T$, and distinguishes if the rule operator $o_1$ meets the commutative property or not and, in contrast, the latter two cases are met when $u_2 \to o_2 r_{21} r_{22}$ and $u_1 \in T$, respectively.

An efficient algorithm with complexity $O(N * M)$ can be designed using Dynamic Programming (Ristad & Yianilos, 1998) to compute the distance between $A$ and $B$, where $N$ and $M$ stand for the number of rules of $A$ and $B$, respectively. As an example, we show the calculation of the proposed measure using two sample SLPs $A$ and $B$, with initial symbols $A_2$ and $B_1$, respectively (see Equation 5).

$$A = \begin{cases} A_0 \ \rightarrow \ + \ x_1 \ x_1 \\ A_1 \ \rightarrow \ / \ A_0 \ A_0 \\ A_2 \ \rightarrow \ * \ A_1 \ w_1 \end{cases} \qquad B = \begin{cases} B_0 \ \rightarrow \ / \ w_1 \ x_1 \\ B_1 \ \rightarrow \ - \ B_0 \ w_1 \end{cases} \tag{5}$$

Table 1 shows the arrangement of rules $A_i$ of SLP $A$ and $B_j$ of SLP $B$ in columns and rows, respectively. Each cell contains the value of the distance $d(A_i, B_j)$. The table must be filled from top to the bottom and from left to right. As an example, the target value $d(A_2, B_1)$ is computed as follows:

$$d(A_2, B_1) = d(*, -) + \min\{d(A_1, B_0) + d(w_1, w_1), d(A_1, w_1) + d(w_1, B_0)\} = 6 \tag{6}$$

| | $\epsilon$ | $x_1$ | $w_1$ | $A_0$ | $A_1$ | $A_2$ |
|---|---|---|---|---|---|---|
| $\epsilon$ | 0 | 1 | 1 | 3 | 7 | 9 |
| $x_1$ | 1 | 0 | 1 | 2 | 6 | 8 |
| $w_1$ | 1 | 1 | 0 | 3 | 7 | 8 |
| $B_0$ | 3 | 3 | 2 | 2 | 5 | 7 |
| $B_1$ | 5 | 5 | 4 | 5 | 6 | 6 |

Table 1: Example of calculation of d(A,B)

The proposed measure $d(A, B)$ is a metric. To prove such statement, we follow the same reasoning that was used in (Waterman et al., 1976) for the edit distance, although specified for SLPs. For this reason, we must first provide some prior definitions.

**Definition I *($\tau$ space).*** Let $Q : V \cup T \cup \{\epsilon\} \rightarrow V \cup T \cup \{\epsilon\}$ be a transformation of a grammar symbol into another, plus the empty word. We define $\tau = \{Q\}$, i.e. the set of all possible transformations of symbols, including identity transformation $I$. As $V \cup T \cup \{\epsilon\}$ is finite, then $\tau$ is finite and every transformation in $\tau$ can be numbered as $\tau = \{Q_1, Q_2, Q_3, ...\}$, and contains all possible insertions, substitutions and deletions over the grammar symbols. Each transformation $Q_i$ has an associated weight $w(Q_i)$. In our case, all weights $w(Q_i) = 1$ except for the identity,

$w(I) = 0$, according to equation 2.

**Definition II *(Transformation sequence).*** Suppose a SLP $A$ and its $j$-th rule $U_j \to o_j r_{j1} r_{j2}$ and a transformation $Q \in \tau$. We define $Q^{j,0}(A) = U_j \to Q(o_j) r_{j1} r_{j2}$, $Q^{j,1} = U_j \to o_j Q(r_{j1}) r_{j2}$, $Q^{j,2} = U_j \to o_j r_{j1} Q(r_{j2})$, i.e. $Q^{j,i}$ is the use of transformation $Q$ at the $i$-th symbol of the consequent of the $j$-th rule.

We define a transformation sequence over a SLP $A$ as $\bar{Q}(A) = (Q^{j_l,i_l}_{k_l} \circ Q^{j_{l-1},i_{l-1}}_{k_{l-1}} \circ ... \circ Q^{j_1,i_1}_{k_1})(A) = Q^{j_l,i_l}_{k_l}(Q^{j_{l-1},i_{l-1}}_{k_{l-1}}(...(Q^{j_1,i_1}_{k_1}(A))...))$. Each transformation sequence $\bar{Q}$ has also a weight, that is calculated as $w(\bar{Q}) = \sum_{p=1}^{l} w(Q^{j_p,i_p}_{k_p})$.

Finally, we define $\{A \to B\}_\tau = \{\bar{Q}(A) : \bar{Q}(A) = B\}$, i.e. the set of sequences of transformations in $\tau$ that transform SLP $A$ into SLP $B$.

**Definition III *(Equivalence relation =).*** Let $SLP$ be the set of all possible SLPs, and $A, B \in SLP$ two SLPs with starting symbols $U^A_N, U^B_M$ respectively, and rules $A = \{U^A_i \to o^A_i r^A_{i1} r^A_{i2}\}$ and $B = \{U^B_j \to o^B_j r^B_{j1} r^B_{j2}\}$, where $o^A_i, o^B_j \in O_u \cup O_b; r^A_{i1}, r^A_{i2}, r^B_{j1}, r^B_{j2} \in V \cup T \cup \{\epsilon\}$. We write the subset of binary operators with commutative property as $C \subseteq O_b$. We define the equivalence relation $= (A, B)$, which we write as $A = B$, as follows:

$A = B \Leftrightarrow U^A_N$ and $U^B_M$ generates the same word or $\exists i, j : 1 \leq i \leq N, 1 \leq j \leq M \wedge o^A_i, o^B_j \in C$ such as the change of operands in the rules $\{U^A_i \to o^A_i r^A_{i2} r^A_{i1}\}$ and/or $\{U^B_j \to o^B_j r^B_{j2} r^B_{j1}\}$ make $U^A_N$ and $U^B_M$ generate the same word. We remark that the existence of rules $i$ and/or $j$ does not have to be unique.

It is easy to verify that $A = A \forall A \in SLP$ (reflexivity), $A = B \Leftrightarrow B = A \forall A, B \in SLP$ (symmetry) and, if $A = B$ and $B = C$ then $A = C, A, B, C \in SLP$ (transitivity), and therefore $=$ is an equivalence relation.

The defined equivalence relation states that two SLPs that provide two algebraic expressions for a SR problem are considered equivalent if both expressions are the same even if their syntax is different due to the effect of commutative operators. Also, $=$ partitions the space into a set of equivalence classes $SLP/=$, where all SLPs that belong to the same class are equivalent under $=$.

**Theorem:** Let $SLP$ be the set of all possible SLPs, and $A, B \in SLP$. The proposed similarity measure $d(A, B)$ is a metric over the quotient space $SLP/=$.

**Proof:** If $A, B, C$ are three different SLPs, and $d(A, B)$ is a metric, then the following conditions must be met (Giles et al., 1987):

$$\begin{aligned}
&d(A,B) \geq 0, && \text{(non negativity)}\\
&d(A,B) = 0 \iff A = B, && \text{(identity)}\\
&d(A,B) = d(B,A) && \text{(symmetry),}\\
&d(A,C) \leq d(A,B) + d(B,C) && \text{(triangle inequality)}
\end{aligned}$$

Proof of conditions of non negativity and identity are trivial from Equations 2 and 3, since $d(A,B)$ is not allowed to have negative values, and two SLPs have $d(A,B) = 0$ only if $A$ and $B$ belong to the same equivalence class. Condition of symmetry is also derived directly from the symmetric property of the equivalence relation $=$ in Definition III.

Regarding the triangle inequality condition, let us rewrite that the distance between the SLPs $A$ and $B$ is computed as the weight of the transformation sequence with minimum number of transformations in $\tau$ as $d(A,B) = \min_{\{A\rightarrow B\}_\tau} \sum_{p=1}^{l_1} w(Q_{k_p}^{j_p i_p})$.

Then, $d(B,A) = \min_{\{B\rightarrow A\}_\tau} \sum_{p=1}^{l_2} w(Q_{k_p}^{j_p i_p})$. As every transformation $Q$ in set $\tau$ has an inverse $Q^{-1}$, (a deletion for an insertion and viceversa, and an inverse transformation for a substitution), if $d(A,B)$ is minimum then $d(A,B) = d(B,A) = \min_{\{A\rightarrow B\}_\tau} \sum_{p=1}^{l_1} w(Q_{k_p}^{j_p i_p}) = \min_{\{B\rightarrow A\}_\tau} \sum_{p=1}^{l_1} w((Q_{k_p}^{j_p i_p})^{-1})$. Also, the distance from $A$ and $B$ to a third SLP $C$ can be written as $d(A,C) = \min_{\{A\rightarrow C\}_\tau} \sum_{p=1}^{l_3} w(Q_{k_p}^{j_p i_p})$ and $d(B,C) = \min_{\{B\rightarrow C\}_\tau} \sum_{p=1}^{l_4} w(Q_{k_p}^{j_p i_p})$.

As every transformation weight is unitary, except for the identity for which $w(I) = 0$, then:

$$\min_{\{A\rightarrow B\}_\tau} \sum_{p=1}^{l_1} w(Q_{k_p}^{j_p i_p}) \quad + \quad \min_{\{B\rightarrow C\}_\tau} \sum_{p=1}^{l_4} w(Q_{k_p}^{j_p i_p}) \quad \geq \quad \min_{\{A\rightarrow C\}_\tau} \sum_{p=1}^{l_3} w(Q_{k_p}^{j_p i_p}) \tag{7}$$

Meaning that the number of steps with unitary weight to transform $A$ into $B$ and then $B$ into $C$ must be greater or equals than the number of steps with unitary weight to transform $A$ into $C$ directly, and then

$$d(A,B) + d(B,C) \geq d(A,C)$$

The opposite condition cannot hold, since all weights are unitary and negative weights are not allowed in the defined distance, so that
$\min_{\{A\rightarrow C\}_\tau} \sum_{p=1}^{l_3} w(Q_{k_p}^{j_p i_p}) < d(A,B) + d(B,C)$ is not possible according to Equations 2 and 3. This concludes with the proof.

## 4. Control of diversity of Straight Line Programs evolution with CHC

In this section we describe an application of the proposed metric to control diversity in Genetic Programming using SLPs as representation for symbolic regression problems. More specifically, we use the proposed distance as a diversity measure in an adapted CHC evolutionary algorithm as incest prevention mechanism. We selected the CHC algorithm since it is a classic approach that combines a balance in diversity and convergence and it has been widely tested in the literature. We name our approach as SLP-CHC and it is based on the real-coded CHC adaptation (Eshelman & Schaffer, 1993). As we evolve SLPs instead of real-coded chromosomes, we use the crossover proposed in (Alonso et al., 2008).

The adaptation of classic CHC implementation to our proposal is shown in Algorithm 1. The procedure starts by initializing a population $P(t)$ of $N$ random SLPs at iteration $t = 0$, then each individual is evaluated by using the Mean Square Error (MSE) as fitness measure between the real output data $y$ and the computed $\tilde{y}$ (see Equation 9). The procedure *averageDistance* calculates the distance threshold $Th$ as the average distance between all individuals of the population, as it is shown in Equation 8, where $d$ is the proposed SLP distance, $N$ is the population size and $c_i$, $c_j$ are the $i$-th and $j$-th SLPs in the population. After initialization, the main algorithm repeats until a stopping criterion is fulfilled. In this work, the stopping criterion used is to reach a number of solutions evaluated. Each algorithm iteration encompasses the following steps: *elitist selection*, *SLP recombination*, *solution evaluation* and the *divergence procedure*. Firstly, we build our population of parents by copying all individuals of the current population in random order. After that, the *SLP recombination* operator (Alonso et al., 2008; Rueda et al., 2019) is applied to generate two new offspring if the SLP distance between the candidate parents exceeds the threshold $Th$. Once the recombination operator is used, each offspring is evaluated according to the fitness measure, and the new population for the next iteration $P(t + 1)$ is built with the $N$ best SLPs between parents and offspring.

Before next iteration starts, it is checked if $P(t + 1) = P(t)$. If so, then the threshold $Th$ is decreased by *rate*. The value of *rate* is defined as $rate = d_{max} * T$ (where $d_{max}$ is the maximum distance between two individuals of the population during initialization, and $T$ is a value in the range $(0, 1)$), and controls the convergence speed according to the diversity of the whole population. Afterwards, if the difference threshold $Th$ is less than 0, the *diverge* procedure is triggered: The new population is composed by $(N-1)$ random SLPs and the best SLP found in the previous generation. Then the difference threshold $Th$ is recalculated by using Equation 8.

$$Th = \frac{1}{N(N-1)} \sum_{1\leq i<N} \sum_{i<j\leq N} d(c_i, c_j) \tag{8}$$

$$MSE = \frac{1}{N} \sum_{i=1}^{n} (\tilde{y} - y)^2 \tag{9}$$

## 5. Experimentation

The main goal of this experimentation is to test if the proposed metric together with the CHC algorithm can improve the exploration and exploitation of the Symbolic Regression solution space in Genetic Programming. As no previous works have been devised to measure similarities between SLPs, we compare the approach with classic metrics for tree representation (Ekárt & Németh, 2000, 2002) and Genetic Programming evolution of SLPs with no diversity control. In particular, we want to compare our proposal with methods used in symbolic regression problems that were specifically designed to increase the diversity of the population using genotypic diversity measures, efficiently computed. More specifically, we used as baseline methods the proposals of Ekart et al. (Ekárt & Németh, 2000, 2002). These approaches compute syntactical differences of the population, represented with tree structures, using a metric based on the edit distance and they demonstrated to be able to increase the diversity, achieving equivalent solutions than classical Genetic Programming algorithms. Due to the approaches of Ekart et al. were able to achieve as robust solution as classical GP algorithms as well as to increase diversity, we consider these approaches as baseline methods of GP.

In order to clarify the comparison carried out in this section, we name each approach as follows: **SLP-GA** for Genetic Programming using SLP representation (Rueda et al., 2019); **SDM** for Genetic Programming approach using fitness sharing and tree representation (Ekárt & Németh, 2000); **DDM** for Genetic Programming that also used fitness sharing and adaptive maintenance of diversity (Ekárt & Németh, 2002); and **SLP-CHC** to refer the method proposed in Section 4. Finally, we performed two experiments to test if our proposal has potential over the mentioned baseline methods: the first one is carried out over a set of synthetic data in subsection 5.1 with the

**Algorithm 1** SLP-CHC algorithm

```
Require: $N$, the number of individuals of the population
Require: $T \in (0,1)$
Require: $\bar{x} = \{(x_1, x_2, ..., x_n)\}$ input data for SLP evaluation
Require: $\bar{w} = (w_1, w_2, ..., w_k)$ a set of constant parameters
Require: $\bar{y} = \{y_1\}$ output data for SLP evaluation
Ensure: SLP(1...N) a sequence of rules that encode the algebraic expression of the best individual
  {Initialization of population}
  Initialize P(t)
  Evaluate(P(t), $\bar{w}, \bar{x}, \bar{y}$)
  Set $t = 1$
  Set d=$averageDistance$
  {SLP-CHC procedure}
  while No stopping criterion is fulfilled do
    $t = t + 1$
    select C(t) from P(t-1)
    C'(t)= $SLP\ recombination$(C(t))
    Evaluate(C'(t), $\bar{w}, \bar{x}, \bar{y}$)
    P(t) = elitist selection(P(t-1), C'(t))
    if $P(t) = P(t-1)$ then
      $Th = Th - rate$
      if $Th < 0$ then
        P(t)=$diverge$
        Evaluate(P(t), $\bar{w}, \bar{x}, \bar{y}$)
        Th=$averageDistance$
      end if
    end if
  end while
  return  Best solution of P(t)
```

aim of validating each approach in a controlled environment. After that, in the second experiment (section 5.2), we deal with a real world problem about energy consumption modelling.

### *5.1. Experimentation with Synthetic Data*

#### *5.1.1. Data acquisition and experimental settings*

We use 19 benchmark algebraic expressions (see Equations 10 to 29) that are widely used in the literature (Nicolau et al., 2015). For each algebraic expression, we generated 500 random data in the domain [0.0,1.0] for each input variable. Finally, each dataset was randomly divided into train (70% of data) and test (remaining 30%). We used the training data to evolve each algorithm to find the best solution and the test data were used to validate each approach. Therefore, results in Subsection 5.1.2 focus on the test set.

The available operators for the symbolic regression datasets are $\{+, -, *, /, \sin, \cos, \log, \min, \max\}$ in all cases, and the set of constant parameters was set to $\bar{w} = (1, 2, 3)$. We performed a preliminary trial-and-error procedure to find the optimal parameters for each algorithm and the best results were provided with the following parameters: we allowed a set of 31 rules for SLP (**SLP-GA** and **SLP-CHC**) and 31 nodes for trees (**SDM** and **DDM**). The population size was set to 100 individuals and the stopping criterion was having 20000 solutions evaluated. Then, for **SLP-GA** we set the crossover and mutation probabilities as 90% and 10% respectively; for both **SDM** and **DDM** we tuned the niche size ($\sigma = 0.5$), $K = 1$ and the crossover and mutation probabilities to to 70% and 30% respectively. With regards to the **SLP-CHC**, the value $T$ used to compute the decrease rate was tuned to 0.3. The fitness measure used was the mean square error (MSE), to be minimized. Finally, we performed 30 executions for each algorithm and problem with different random seeds to carry out a statistical test that helped us to determine if there exist significant differences between the results obtained.

$$f_1(x_1, x_2) = \frac{e^{-(x_1-1)^2}}{1.2 + (x_2 - 2.5)^2} \tag{10}$$

$$f_2(x_1, x_2) = e^{-x_1} x_1^3 \cos(x_1) \sin(x_1) * (\cos(x_1) \sin^2(x_1) - 1)(x_2 - 5) \tag{11}$$

$$f_3(x_1, x_2, x_3) = 30 \frac{(x_1 - 1)(x_3 - 1)}{x_2^2 (x_1 - 10)} \tag{12}$$

$$f_4(x_1, x_2) = 6 \sin(x_1) \cos(x_2) \tag{13}$$

$$f_5(x_1, x_2) = (x_1 - 1)(x_2 - 3) + 2\sin((x_1 - 4)(x_2 - 4)) \tag{14}$$

$$f_6(x_1, x_2) = \frac{(x_1 - 3)^4 + (x_2 - 3)^3 - (x_2 - 3)}{(x_2 - 2)^4 + 10} \tag{15}$$

$$f_7(x_1, x_2) = \frac{1}{1 + x_1^{-4}} + \frac{1}{1 + x_2^{-4}} \tag{16}$$

$$f_8(x_1, x_2) = x_1^4 - x_1^3 + \frac{x_2^2}{2} - x_2 \tag{17}$$

$$f_9(x_1, x_2) = \frac{8}{2 + x_1^2 + x_2^2} \tag{18}$$

$$f_{10}(x_1, x_2) = \frac{x_1^3}{5} + \frac{x_2^3}{2} - x_2 - x_1 \tag{19}$$

$$f_{11}(x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}) = x_1 x_2 \tag{20}$$

$$+x_3 x_4 + x_5 x_6 + x_1 x_7 x_9 + x_3 x_6 x_{10} \tag{21}$$

$$f_{12}(x_1, x_2, x_3, x_4, x_5) = -5.41 + 4.9\frac{x_4 - x_1 + \frac{x_2}{x_5}}{3x_4} \tag{22}$$

$$f_{13}(x_1, x_2, x_3, x_4, x_5, x_6) = \frac{(x_5 x_6)}{\frac{x_1}{x_2}\frac{x_3}{x_4}} \tag{23}$$

$$f_{14}(x_1, x_2, x_3, x_4, x_5) = 0.81 + 24.3\frac{2x_2 + 3x_3^2}{4x_4^3 + 5x_5^4} \tag{24}$$

$$f_{15}(x_1, x_2, x_3, x_4, x_5) = 32 - 3\frac{\tan(x_1)}{\tan(x_2)}\frac{\tan(x_3)}{\tan(x_4)} \tag{25}$$

$$f_{16}(x_1, x_2, x_3, x_4, x_5) = 22 - 4.2(\cos(x_1) - \tan(x_2)) * (\frac{\tanh(x_3)}{\sin(x_4)}) \tag{26}$$

$$f_{17}(x_1, x_2, x_3, x_4, x_5) = x_1 x_2 x_3 x_4 x_5 \tag{27}$$

$$f_{18}(x_1, x_2, x_3, x_4, x_5) = 12 - 6\frac{\tan(x_1)}{e^{x_2}}(x_3 - \tan(x_4)) \tag{28}$$

$$f_{19}(x_1, x_2, x_3, x_4, x_5) = 2 - 2.1\cos(9.8x_1)\sin(1.3x_5) \tag{29}$$

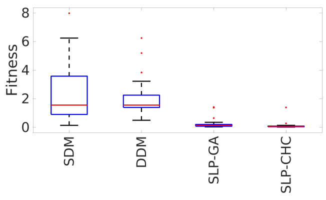
### *5.1.2. Results and discussion*

The results obtained in the test datasets for each approach are shown in Table 2. Each row is associated with the results of its corresponding benchmark dataset (Column 1). Then, for each algorithm we show the median MSE and the average execution time measured in seconds. We remark that we use the median value instead of the mean since the results do not follow a normal distribution and, in these cases, the mean cannot be considered an appropriate statistic summarization value. We have also included boxplots with the MSE distributions in the test sets to give support to our analysis in Figure 1, and they include information about the best and worst solutions.

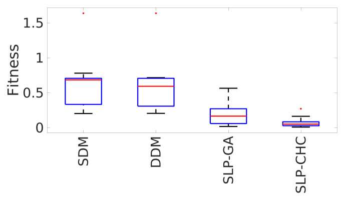
| | **SDM** | | **DDM** | | **SLP-GA** | | **SLP-CHC** | |
|---|---|---|---|---|---|---|---|---|
| | **Test** | **Time** | **Test** | **Time** | **Test** | **Time** | **Test** | **Time** |
| $f_1$ | 1.55 (3) | 2.9 | 1.53 (3) | 2.5 | 0.12 (2) | 2.43 | $4.14 \times 10^{-2}$ (1) | 4.2 |
| $f_2$ | 0.68 (3) | 2.73 | 0.59 (3) | 2.26 | 0.16 (2) | 2.93 | $5.33 \times 10^{-2}$ (1) | 5.46 |
| $f_3$ | $7.93 \times 10^{6}$ (2) | 3.03 | $7.94 \times 10^{6}$ (2) | 2.4 | $4.35 \times 10^{6}$ (1) | 2.36 | $4.59 \times 10^{6}$ (1) | 4.16 |
| $f_4$ | 1.47 (3) | 2.9 | 1.46 (3) | 2.6 | 0.21 (2) | 2.83 | $6.27 \times 10^{-2}$ (1) | 5.3 |
| $f_5$ | 8.01 (3) | 2.96 | 7.42 (3) | 2.93 | 1.9 (2) | 2.7 | 0.89 (1) | 4.6 |
| $f_6$ | 2.25 (3) | 3 | 2.25 (3) | 3 | 0.18 (2) | 2.5 | $8.22 \times 10^{-2}$ (1) | 4.9 |
| $f_7$ | $1.44 \times 10^{-2}$ (3) | 3 | $1.51 \times 10^{-2}$ (3) | 2.9 | $1.39 \times 10^{-2}$ (2) | 2.73 | $1.33 \times 10^{-2}$ (1) | 4.93 |
| $f_8$ | $2.49 \times 10^{-2}$ (3) | 2.96 | $2.49 \times 10^{-2}$ (3) | 2.93 | $6.92 \times 10^{-3}$ (2) | 4 | $1.8 \times 10^{-3}$ (1) | 5.33 |
| $f_9$ | 0.24 (3) | 2.33 | 0.25 (3) | 2.36 | $8.21 \times 10^{-2}$ (2) | 2.93 | $1.99 \times 10^{-2}$ (1) | 5.6 |
| $f_{10}$ | $3.45 \times 10^{-2}$ (3) | 2.76 | $3.45 \times 10^{-2}$ (3) | 2.73 | $1.47 \times 10^{-2}$ (2) | 3.2 | $3.99 \times 10^{-2}$ (1) | 4.96 |
| $f_{11}$ | 0.23 (3) | 3 | 0.22 (3) | 2.83 | 0.15 (2) | 3 | 0.14 (1) | 9.33 |
| $f_{12}$ | $3.99 \times 10^{4}$ (2) | 2.96 | $3.99 \times 10^{4}$ (2) | 2.96 | $1.113 \times 10^{4}$ (1) | 2.5 | $2.15 \times 10^{4}$ (1) | 5.96 |
| $f_{13}$ | $4.91 \times 10^{2}$ (2) | 3.03 | $4.89 \times 10^{2}$ (2) | 3 | $1.25 \times 10^{2}$ (1) | 2.73 | $1.19 \times 10^{2}$ (1) | 6.33 |
| $f_{14}$ | $6.99 \times 10^{7}$ (3) | 2.96 | $6.99 \times 10^{7}$ (3) | 2.96 | $1.65 \times 10^{7}$ (1) | 2.63 | $2.6 \times 10^{7}$ (2) | 5.1 |
| $f_{15}$ | $3.77 \times 10^{4}$ (3) | 2.76 | $3.8 \times 10^{4}$ (3) | 2.86 | $2.28 \times 10^{4}$ (2) | 2.43 | $8.53 \times 10^{3}$ (1) | 5.43 |
| $f_{16}$ | $8.81 \times 10^{2}$ (1) | 3 | $8.82 \times 10^{2}$ (1) | 2.96 | $9.42 \times 10^{2}$ (1) | 2.53 | $8.84 \times 10^{2}$ (1) | 5.16 |
| $f_{17}$ | $4.76 \times 10^{-3}$ (2) | 2.2 | $4.76 \times 10^{-3}$ (2) | 2.73 | $2.57 \times 10^{-3}$ (1) | 2.63 | $1.79 \times 10^{-3}$ (1) | 5.6 |
| $f_{18}$ | 0.6 (3) | 2.9 | 0.55 (3) | 2.86 | 1.42 (2) | 2.53 | 1.26 (1) | 4.63 |
| $f_{19}$ | 0.86 (3) | 2.96 | 0.86 (3) | 2.7 | 0.81 (2) | 2.9 | 0.73 (1) | 4.66 |

Table 2: Results of **SDM**, **DDM**, **SLP-GA** and **SLP-CHC** in benchmark algebraic expressions

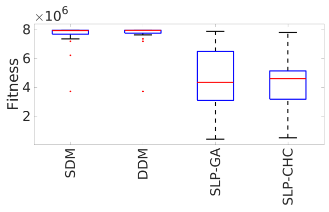
The Shapiro-Wilk test has been applied to check if the results obtained for each approach follow normality conditions. As the fitness distribution results did not follow a normal distribution, we performed a non-parametric Kruskal-Wallis test (KW) with a 95% of confidence level to validate if there are significant differences between each approach statistically. The results of the KW test are presented together with the median fitness result of each approach in Columns 2, 4, 6 and 8 in Table 2, in brackets. More specifically, we performed a ranking between each approach and benchmark algebraic expression from 1 (the best approach) to 4 (worst algorithm results). If there are no significant differences between the two algorithms in a dataset, they are ranked with the



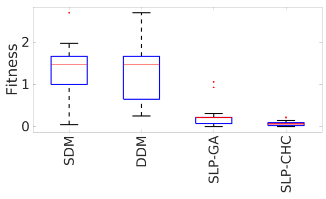
(a) Benchmark algebraic expression $f_1$



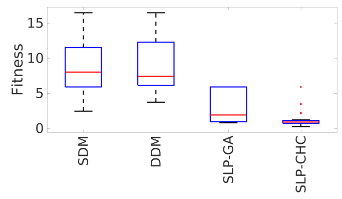
(b) Benchmark algebraic expression $f_2$



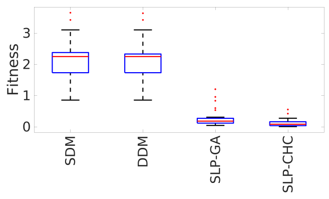
(c) Benchmark algebraic expression $f_3$



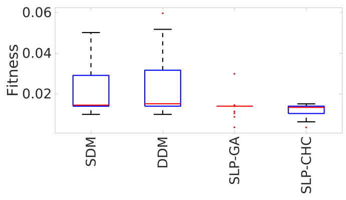
(d) Benchmark algebraic expression $f_4$



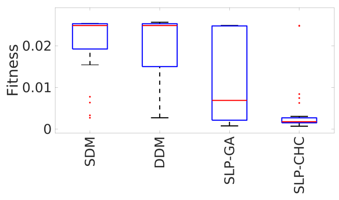
(e) Benchmark algebraic expression $f_5$



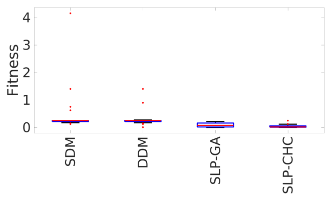
(f) Benchmark algebraic expression $f_6$



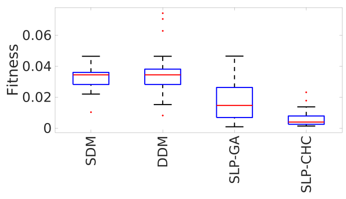
(g) Benchmark algebraic expression $f_7$



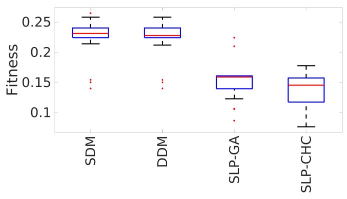
(h) Benchmark algebraic expression $f_8$



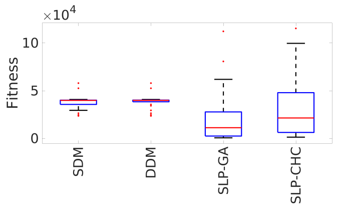
(i) Benchmark algebraic expression $f_9$



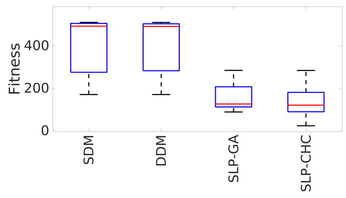
(j) Benchmark algebraic expression $f_{10}$



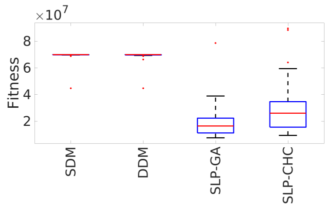
(k) Benchmark algebraic expression $f_{11}$



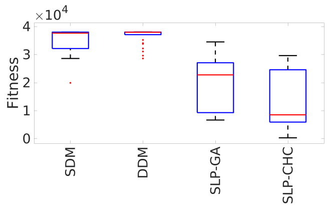
(l) Benchmark algebraic expression $f_{12}$



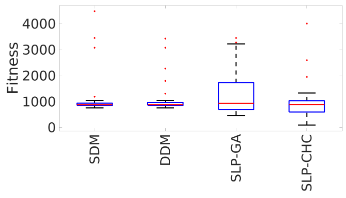
(m) Benchmark algebraic expression $f_{13}$



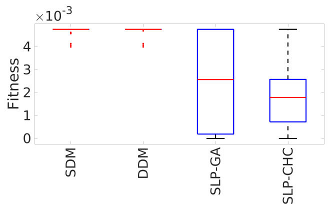
(n) Benchmark algebraic expression $f_{14}$



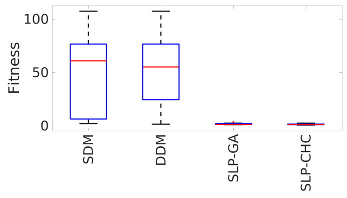
(o) Benchmark algebraic expression $f_{15}$



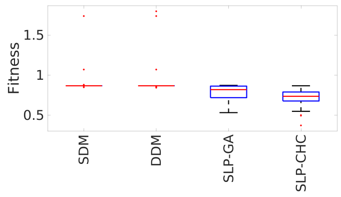
(p) Benchmark algebraic expression $f_{16}$



(q) Benchmark algebraic expression $f_{17}$



(r) Benchmark algebraic expression $f_{18}$



(s) Benchmark algebraic expression $f_{19}$

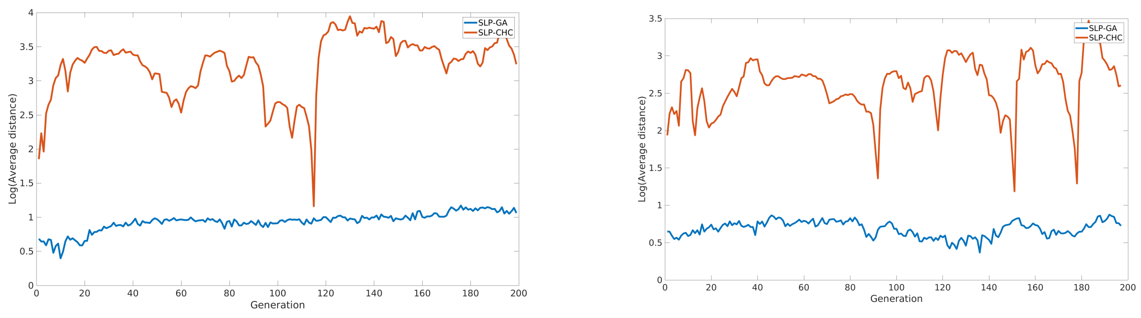


Figure 2: Diversity of SLP-GA (blue line) and SLP-CHC (red line) calculated as the average distance measure of the population

same number.

In a first analysis of the results, we start with a comparison of the selected baseline methods (**SDM** and **DDM**). Results of the applied statistical test suggest that there are no differences regarding fitness in both approaches. This is consistent with the work of Ékart and Németh (Ekárt & Németh, 2002), where they argued that diversity is increased using their method, but they did not find improvement in the fitness of solutions.

In a previous work (Rueda et al., 2019) we discussed how the SLP representation may overcome traditional limitations of tree representation. In the experimentation performed, this is validated and **SLP-GA** improves **SDM** and **DDM** in 18 of 19 problems studied, according to Table 2 and Figure 1. The remaining of the analysis focuses on the comparison of **SLP-CHC** and **SLP-GA**, consequently.

We continue our analysis by comparing the results of **SLP-GA** and **SLP-CHC** with the aim of verifying our initial hypothesis that the proposed metric together with the CHC algorithm helps to improve the control of diversity and convergence and also overcomes local optima. **SLP-CHC** has provided better results than **SLP-GA** in 13 cases, **SLP-CHC** and **SLP-GA** were equivalent in 5 cases, and **SLP-GA** was the best algorithm in 1 case. This confirms that the proposed metric can be used as a diversity measure in Genetic Programming to find a balance between exploration and exploitation.

Regarding the best solutions found by **SLP-GA** and **SLP-CHC**, Figure 1 shows that **SLP-CHC** provided the lowest fitness in 14 cases and **SLP-GA** achieved the lowest fitness in 5 experiments. With regards to the worst fitness value, **SLP-GA** provided the worst solution in 13 cases, meanwhile **SLP-CHC** did it in 4 experiments. In the remaining 2 cases, **SLP-GA** and **SLP-CHC** obtained the same worst solution.

Regarding the algorithm robustness, Figure 1 also shows that **SLP-CHC** is more robust than **SLP-GA**, since the distance between the intermediate quartiles in the boxplots are lower in 14 cases for **SLP-CHC**, meaning that it is expected that a random execution of **SLP-CHC** will provide better results than **SLP-GA**.

The discussion follows with a diversity study of the results provided for both **SLP-GA** and **SLP-CHC**. We selected the executions that provided the best SLP for each approach, with the aim of analyzing the diversity behavior during the evolutionary process. Figure 2 describes a sample for the problems $f_{14}, f_{16}$, where the axis $X$ stands for the generation of each approach and the axis $Y$ for the average diversity in log scale, calculated as described in Equation 8. Blue lines represent the diversity measured for **SLP-GA** and red lines stand for the diversity of **SLP-CHC**.

We may see that the **SLP-GA** approach was able to preserve diversity during the evolutionary cycle, since the average distance was not decreasing during all generations. However, it was unable to explore the solution space enough to overcome local optima. Nevertheless, the **SLP-CHC** approach increased diversity in the population substantially, leading to a better exploration of the search space. In the cases when the population diversity decreased, the diverge procedure allowed to explore unknown areas since different individuals were included in the population. These facts suggest us that an increase of population diversity may help to reduce the premature convergence, allowing **SLP-CHC** to overcome the results of **SLP-GA**.

We conclude with this section with an analysis of the execution time. We can see in Table 2 that **SLP-CHC** was computationally more expensive than the remaining baseline approaches. This fact is a consequence of both representation schemes used to encode individuals and the similarity measure used in each approach. Whereas the metric used in **SDM** and **DDM** does not take into account commutative operations and only compare two trees node by node, our proposed similarity measure used in **SLP-GA** and **SLP-CHC** takes into account the grammar that generates an algebraic expression, as well as commutative property of operators. Thus, an increase in the execution time is expected, since for each pair of parents it is required to calculate the distance before crossover is applied. However, the increase in computational time observed could be palliated with the benefits of applying diversity control using the proposed metric, depending on the researcher needs to avoid local optima.

## *5.2. Experimentation with Real Data*

### *5.2.1. Data Acquisition and experimental settings*

With the approaches validated in a controlled environment, this section describes the experimentation with real data. The data used in this experimentation were provided from a set of energy consumption time series of different buildings coming from the University of Granada from March 2013 to October 2015, hourly measured. More specifically, the buildings are two research centres and two faculties, which we named as $B_1$, $B_2$, $B_3$ and $B_4$ for confidentiality reasons. The energy consumption of these buildings are shown in Figure 3, where the axis $X$ stands for the time and the axis $Y$ for the energy consumption in kW/h. Before using the data, it must be preprocessed because could be missing data due to sensor failures or light cuts. The preprocessing step encompasses an interpolation of the missing values (around a 5% of the data) and a time alignment to obtain the data in the same temporal range. Finally, we aggregate the energy (kW) consumed each 24 hours of the same day to get a final dataset. Our initial hypothesis in this experimentation is that the energy consumption of a weekday can be modelled as a combination of the energy consumption of the remaining working days in the same week. Consequently, we carried out a correlation analysis of the energy consumption of each weekday and building with the aim of understanding the energy consumption behavior. To that end, Figures 4 to 7 show the correlation plot matrices for each working day and building. In each correlation plot, the diagonal shows the histogram, which provides us information about the energy consumption distribution for each working day. Moreover, the text red in the remaining scatter plots gathers the correlation coefficient R, ranging from -1 to 1, between the days of the corresponding row and column of the plot matrices. The mentioned correlation coefficient denotes whether exists a positive or negative correlation between two variables. Values near to 1 denotes high positive correlation (respectively to -1 with high negative correlation), meanwhile values closer to 0 mean low correlation. In this way, the mentioned figures verify that there is a high ($R > 0.7$) or medium ($0.3 \leq R \leq 0.7$) positive correlation between the energy consumption of the working days.

In the experiments, we divided each dataset into train (70%) and test (30%) data to avoid over-fitting. The train set was used to build each model of each approach and then it was tested over the test set to validate the quality of the solutions found. Moreover, as well as in synthetic

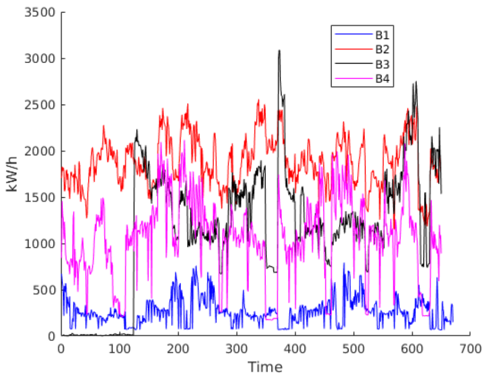


Figure 3: Energy consumption data series of buildings $B_1$, $B_2$, $B_3$ and $B_4$

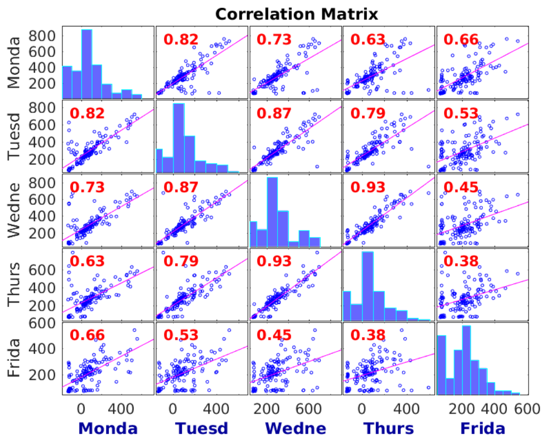


Figure 4: Energy consumption correlation between working days for building $B_1$

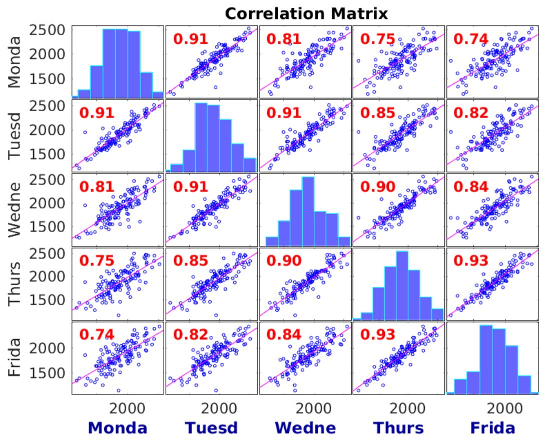


Figure 5: Energy consumption correlation between working days for building $B_2$

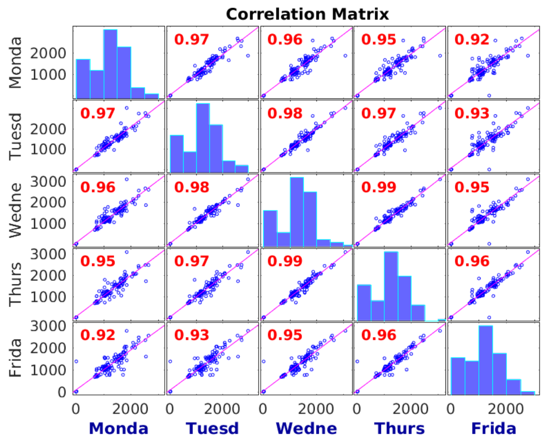


Figure 6: Energy consumption correlation between working days for building $B_3$

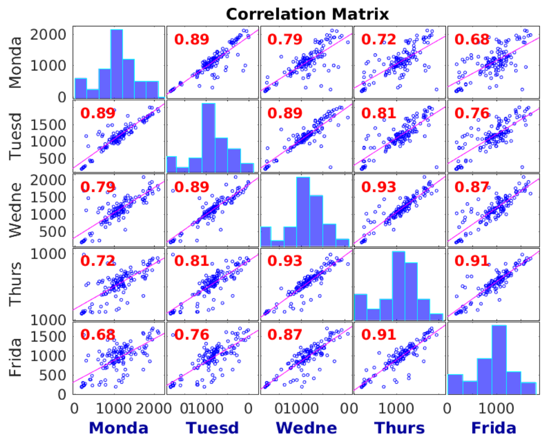


Figure 7: Energy consumption correlation between working days for building $B_4$

data experiments, we performed 30 executions for each approach and problem with different seed to perform statistical tests. Although we kept the same configuration for symbolic regression (parameters $\bar{w}$ and mathematical operators), we performed a preliminary trial-and-error procedure to find the optimal parameter for each approach, and the best results were obtained with the following parameters: we allow a set of 31 rules for SLP and 31 nodes for trees. The population size was set to 200 individuals and the stopping criterion was 20000 solutions evaluated. Then, for *SLP-GA* we tuned the crossover and mutation probabilities to 90% and 10% respectively. For **SDM** and **DDM** we set the crossover and mutation probabilities to 80% and 20% respectively, and the niche size ($\sigma$) to 0.05. Regarding the **SLP-CHC** approach, the value used to compute the decrease rate was set to 0.3.

*5.2.2. Results and Discussion*

Table 3 shows the results obtained for each approach and problem in test data. The columns are organized in groups of two, for each algorithm analyzed (**SDM**, **DDM**, **SLP-GA** and **SLP-CHC**). For each approach, we focus on the measurements *Test* which contains the median MSE of 30 runnings and *Time* that gathers the average time spent by the algorithm in the 30 runnings, measured in milliseconds. Then, the rows are organized in groups of 5, where each group references the working day of each building ($B_1$ to $B_4$). On the other hand, to provide a better analysis of the results, we included boxplots of the MSE distribution in all experiments in Figure 8. Each figure contains the boxplots of the MSE for the algorithms being compared, i.e. **SDM**, **DDM**, **SLP-GA** and **SLP-CHC**, for the same building and working day. Besides, we carried out a statistical test to empirically validate if one approach has potential over them. We first performed a normality test (Shapiro-Wilk test) to verify if the results provided for each approach follow a normal distribution. As the results did not follow a normal distribution, we performed a non-parametric Kruskal-Wallis test (KW) with a 95% confidence level. The results of the KW test were presented together with the median results of each approach in brackets (Columns 2, 4, 6 and 8) as a sorted list that comes from 1 to 4, where algorithms marked to 1 mean that were better than the algorithms marked with 2, 3 and 4 respectively. If two approaches were marked with the same number meant that no significant differences were found between each approach.

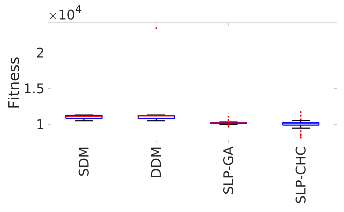
Similarly to the experimentation in benchmark data and the results of the work (Ekárt & Németh, 2002), the baseline methods (**SDM** and **DDM**) did not present significant differences in the results over real energy consumption data in terms of accuracy. Moreover, regarding the results of Table 3 and the boxplots of Figure 8, SLP approaches overcame the results of **SDM** and **DDM** in 15 of 20 problems.

Therefore, we focus the analysis of this experimentation on the results of **SLP-GA** and **SLP-CHC**. Regarding the results of the median values of **SLP-GA** and **SLP-CHC** of Table 3 and the results of the second quartile of the boxplots in Figure 8, we may verify that **SLP-CHC** provided better results in 17 of 20 problems while **SLP-GA** did it in the remaining 3 experiments. With regards to the KW results, **SLP-CHC** proved to be significantly better in 10 of 20 experiments, meanwhile in the remaining 10 experiments no significant differences were found. Regarding the execution time, similar conclusion to Section 5.1 may be achieved. Although the proposed similarity measure together with the CHC adaptation helped to avoid local optima and performed better results, it caused an increase of the computational time of **SLP-CHC**, being more computational expensive than the remaining approaches.

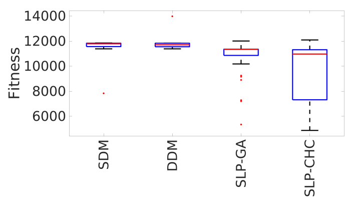
| | **SDM** | | **DDM** | | **SLP-GA** | | **SLP-CHC** | |
|---|---|---|---|---|---|---|---|---|
| | **Test** | **Time** | **Test** | **Time** | **Test** | **Time** | **Test** | **Time** |
| | Building $B_1$ | | | | | | | |
| **Monday** | $1.11 \times 10^4$ (2) | $1.63 \times 10^3$ | $1.12 \times 10^4$ (2) | $1.33 \times 10^3$ | $1.02 \times 10^4$ (1) | $6.4 \times 10^2$ | $1.01 \times 10^4$ (1) | $2.86 \times 10^3$ |
| **Tuesday** | $1.18 \times 10^4$ (3) | $1.33 \times 10^3$ | $1.17 \times 10^4$ (3) | $1.07 \times 10^3$ | $1.13 \times 10^4$ (2) | $6.32 \times 10^2$ | $1.1 \times 10^4$ (1) | $2.99 \times 10^3$ |
| **Wednesday** | $1.18 \times 10^4$ (3) | $1.37 \times 10^3$ | $1.17 \times 10^4$ (3) | $1.07 \times 10^3$ | $1.13 \times 10^4$ (2) | $6.43 \times 10^2$ | $1.12 \times 10^4$ (1) | $2.69 \times 10^3$ |
| **Thursday** | $1.05 \times 10^4$ (2) | $1.43 \times 10^3$ | $1.03 \times 10^4$ (2) | $1.03 \times 10^3$ | $7.72 \times 10^3$ (1) | $6.51 \times 10^2$ | $7.62 \times 10^3$ (1) | $2.85 \times 10^3$ |
| **Friday** | $1.05 \times 10^4$ (3) | $1.2 \times 10^3$ | $1.04 \times 10^4$ (3) | $1.1 \times 10^3$ | $7.77 \times 10^3$ (2) | $6.59 \times 10^2$ | $7.64 \times 10^3$ (1) | $2.75 \times 10^3$ |
| | Building $B_2$ | | | | | | | |
| **Monday** | $7.26 \times 10^3$ (2) | $1.53 \times 10^3$ | $7.23 \times 10^3$ (2) | $1.4 \times 10^3$ | $5.71 \times 10^3$ (1) | $6.34 \times 10^2$ | $4.63 \times 10^3$ (1) | $3.19 \times 10^3$ |
| **Tuesday** | $2.33 \times 10^3$ (2) | $1.4 \times 10^3$ | $2.28 \times 10^3$ (2) | $1.33 \times 10^3$ | $2.21 \times 10^3$ (1) | $6.81 \times 10^2$ | $2.02 \times 10^3$ (1) | $2.68 \times 10^3$ |
| **Wednesday** | $3.95 \times 10^3$ (3) | $1.33 \times 10^3$ | $3.94 \times 10^3$ (3) | $1.27 \times 10^3$ | $3.48 \times 10^3$ (2) | $6.46 \times 10^2$ | $3.28 \times 10^3$ (1) | $2.85 \times 10^3$ |
| **Thursday** | $3.97 \times 10^3$ (3) | $1.47 \times 10^3$ | $3.97 \times 10^3$ (3) | $1.2 \times 10^3$ | $3.49 \times 10^3$ (2) | $7.1 \times 10^2$ | $3.22 \times 10^3$ (1) | $2.63 \times 10^3$ |
| **Friday** | $1.89 \times 10^4$ (3) | $1.77 \times 10^3$ | $1.89 \times 10^4$ (3) | $1.33 \times 10^3$ | $7.04 \times 10^3$ (2) | $6.86 \times 10^2$ | $5.63 \times 10^3$ (1) | $3.03 \times 10^3$ |
| | Building $B_3$ | | | | | | | |
| **Monday** | $2.98 \times 10^3$ (1) | $1.33 \times 10^3$ | $2.98 \times 10^3$ (1) | $1.17 \times 10^3$ | $4.2 \times 10^3$ (2) | $6.45 \times 10^2$ | $4.34 \times 10^3$ (2) | $2.62 \times 10^3$ |
| **Tuesday** | $1.42 \times 10^3$ (1) | $1.43 \times 10^3$ | $1.42 \times 10^3$ (1) | $1.3 \times 10^3$ | $1.52 \times 10^3$ (2) | $7.67 \times 10^2$ | $1.7 \times 10^3$ (2) | $3.2 \times 10^3$ |
| **Wednesday** | $2.56 \times 10^3$ (1) | $1.4 \times 10^3$ | $2.53 \times 10^3$ (1) | $1.17 \times 10^3$ | $2.91 \times 10^3$ (2) | $6.4 \times 10^2$ | $2.92 \times 10^3$ (2) | $2.69 \times 10^3$ |
| **Thursday** | $2.57 \times 10^3$ (1) | $1.23 \times 10^3$ | $2.57 \times 10^3$ (1) | $1.13 \times 10^3$ | $2.93 \times 10^3$ (2) | $6.47 \times 10^2$ | $2.92 \times 10^3$ (2) | $2.51 \times 10^3$ |
| **Friday** | $5.26 \times 10^4$ (3) | $1.47 \times 10^3$ | $5.26 \times 10^4$ (3) | $1.03 \times 10^3$ | $5.22 \times 10^4$ (2) | $7.05 \times 10^2$ | $4.93 \times 10^3$ (1) | $2.93 \times 10^3$ |
| | Building $B_4$ | | | | | | | |
| **Monday** | $3.93 \times 10^4$ (3) | $1.47 \times 10^3$ | $3.94 \times 10^4$ (3) | $1.33 \times 10^3$ | $3.73 \times 10^4$ (2) | $6.4 \times 10^2$ | $3.68 \times 10^4$ (1) | $2.91 \times 10^3$ |
| **Tuesday** | $2.68 \times 10^4$ (3) | $1.43 \times 10^3$ | $2.66 \times 10^4$ (3) | $1.3 \times 10^3$ | $2.49 \times 10^4$ (2) | $6.17 \times 10^2$ | $2.47 \times 10^4$ (1) | $2.79 \times 10^3$ |
| **Wednesday** | $2.68 \times 10^4$ (3) | $1.43 \times 10^3$ | $2.66 \times 10^4$ (3) | $1.37 \times 10^3$ | $2.47 \times 10^4$ (2) | $6.62 \times 10^2$ | $2.36 \times 10^4$ (1) | $2.86 \times 10^3$ |
| **Thursday** | $3.1 \times 10^4$ (1) | $1.37 \times 10^3$ | $3.1 \times 10^4$ (1) | $1.3 \times 10^3$ | $3.14 \times 10^4$ (1) | $6.68 \times 10^2$ | $3.11 \times 10^4$ (1) | $2.52 \times 10^3$ |
| **Friday** | $8.31 \times 10^4$ (2) | $1.6 \times 10^3$ | $8.3 \times 10^4$ (2) | $1.43 \times 10^3$ | $5.33 \times 10^4$ (1) | $7.14 \times 10^2$ | $4.91 \times 10^4$ (1) | $3.01 \times 10^3$ |

Table 3: Results of **SDM**, **DDM**, **SLP-GA** and **SLP-CHC** in real energy consumption data

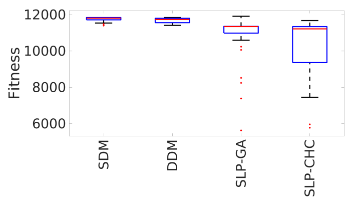
On the other hand, we have also carried out an analysis of the diversity measured during the genetic procedure of both **SLP-GA** and **SLP-CHC** approaches. Figure 9 shows the diversity registered (measured using Equation 8) during the training procedure for both **SLP-GA** and **SLP-CHC** in two scenarios. As can be seen, **SLP-CHC** helped to increase the diversity of the population and, consequently to achieve a better exploration of the search space. The results of Table 3 together with the diversity analysis suggest that **SLP-CHC** has potential over **SLP-GA**.



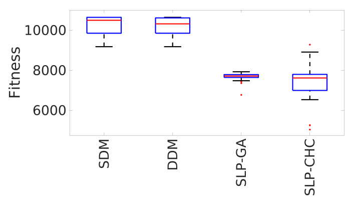
(a) $B_1$ Monday



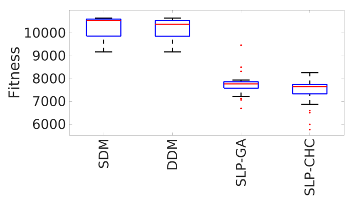
(b) $B_1$ Tuesday



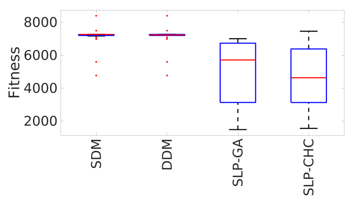
(c) $B_1$ Wednesday



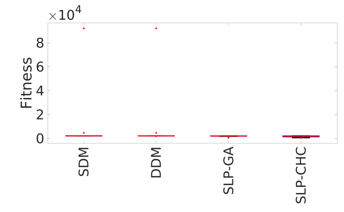
(d) $B_1$ Thursday



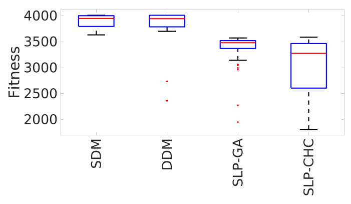
(e) $B_1$ Friday



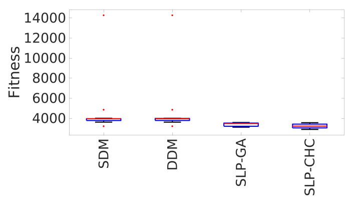
(f) $B_2$ Monday



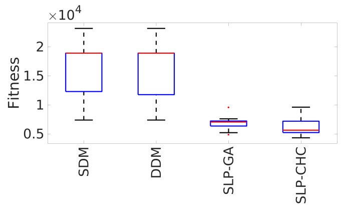
(g) $B_2$ Tuesday



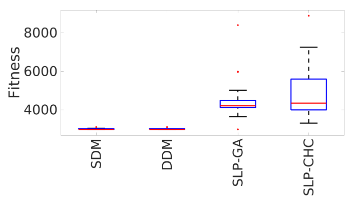
(h) $B_2$ Wednesday



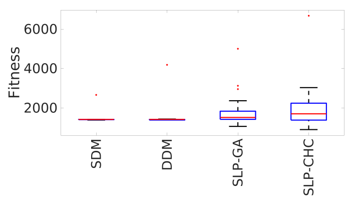
(i) $B_2$ Thursday



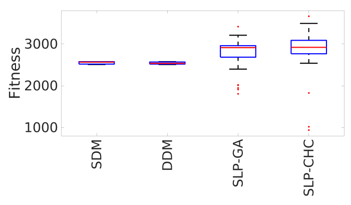
(j) $B_2$ Friday



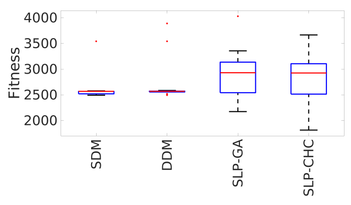
(k) $B_3$ Monday



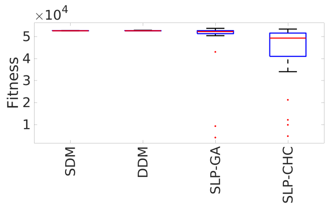
(l) $B_3$ Tuesday



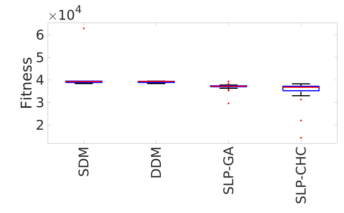
(m) $B_3$ Wednesday



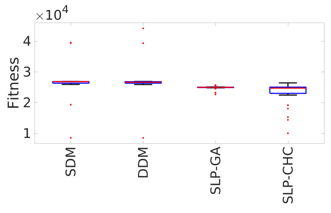
(n) $B_3$ Thursday



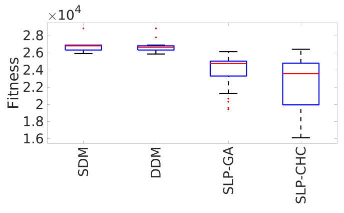
(o) $B_3$ Friday



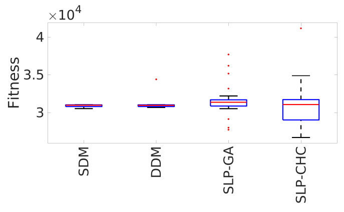
(p) $B_4$ Monday



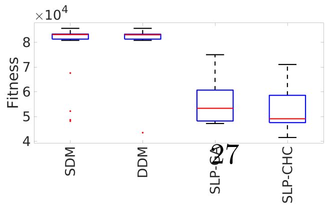
(q) $B_4$ Tuesday



(r) $B_4$ Wednesday

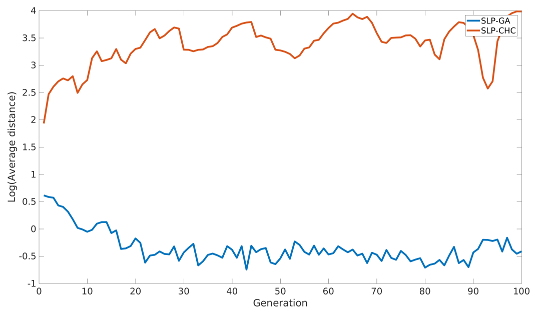


(s) $B_4$ Thursday



(t) $B_4$ Friday

Figure 8: Boxplots of accuracy for SDM, DDM, SLP-GA and SLP-CHC for each building and working day



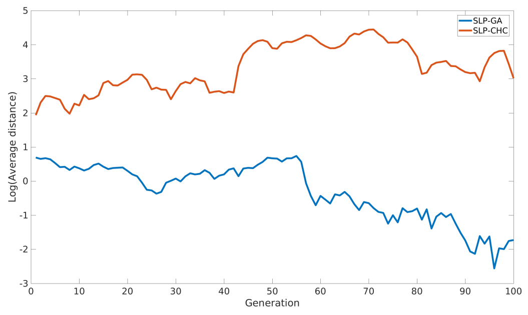
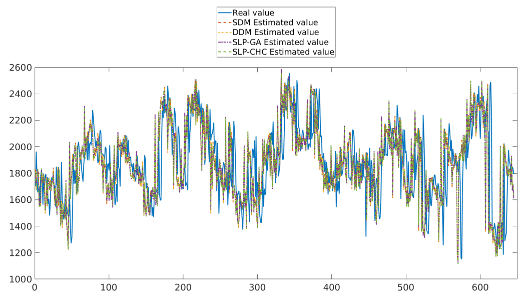


Figure 9: Diversity of SLP-GA (blue line) and SLP-CHC (red line) in energy consumption data on Monday of building $B_2$ (left) and on Thursday of building $B_3$ (right). The diversity was calculated as the average distance measure of the population.

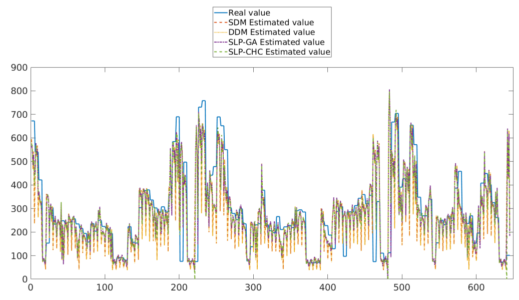
To conclude with the analysis of this experimentation, Figure 10 shows the original datasets and the results of the modelled data for each building. These results help us to conclude that **SLP-CHC** is a promising alternative for real applications of symbolic regression because although the results of Table 3 suggest a high MSE value, the plots verified that the modelled data fits correctly the real data.

## 6. Conclusions

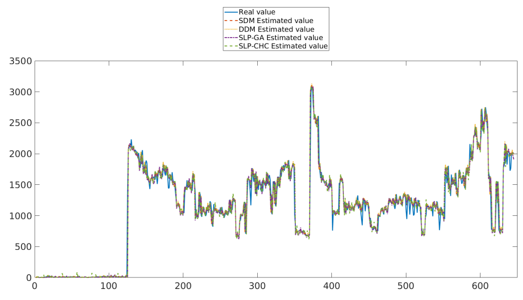
This manuscript has addressed the problem of holding a balance in diversity and convergence in genetic programming, and more specifically symbolic regression. The outcomes of this piece of research encompass both theory and practice. Regarding the theoretical dimension, we have developed a metric that can be used to calculate the structural distance between symbolic regression expressions represented as Straight Line Programs (SLP). Such metric works similarly to the *edit distance*, and it is able to provide a similarity measure between algebraic expressions encoded as SLPs. In addition, we have shown that it is possible to include certain semantic information in the distance computation regarding properties of algebraic operators, such as commutativity. In our opinion, this could be a relevant milestone that can help to develop new techniques that trascend classic distance measurement models that are purely syntax-based, and become a nexus with semantic ones to make a more effective comparison. We presume that the field of symbolic reduction could provide some inspiration to achieve this goal. However, as it is shown in our experimentation, the inclusion of a simple semantic analysis such as commutativity to compare



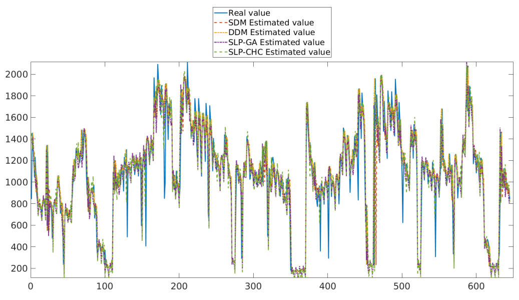
(a) Building $B_1$



(b) Building $B_2$



(c) Building $B_3$



(d) Building $B_4$

Figure 10: Plots of real data (blue), SDM estimated data (red), DDM estimated data (yellow), SLP-GA estimated data (violet) and SLP-CHC estimated data (green) for the buildings $B_1$ to $B_4$

algebraic expressions could lead to higher computational requirements.

Beyond the specific use case of symbolic regression studied in this paper, we believe that the theory behind the methodology used to obtain the metric can be easily expanded to other genetic programming applications, such as computer program generation, grammar evolution, etc. For example, in a near future research work, we plan to use the outcomes shown in this manuscript for automatic generation of quantum algorithms using SLP-encoded genetic programming. We also plan to use the metric design methodology described in this article to tackle the decoherence problem of qubits in large quantum algorithm's circuit design.

In regards to the experimental outcomes, we have adapted the developed metric into the *incest prevention* mechanism of the CHC algorithm, to control diversity and convergence during the evolutionary process of SLP-encoded algebraic expressions. As no previous works define a similarity measure for SLPs, we have compared our approach with other tree-based representation genetic programming proposals existing in the literature. Experiments in both benchmark and real application datasets show that the proposed metric can be used by evolutionary algorithms to increase diversity in the population, and that its integration within the CHC algorithm helps to improve the balance in diversity and convergence. In particular, we tested the approach in a real energy consumption modelling problem, where time series containing energy consumption of public buildings at the University of Granada are provided as input. The results in the real dataset are consistent with the results of the benchmark data, which suggests that the approach has potential and can be extended to different real applications, and more specifically in time series modelling domains.

An additional experimental outcome is that the proposal is able to overcome local optima found by the state-of-the-art approaches. We consider that this fact is specially relevant considering that previous research articles that propose distance measures for tree-based representation concluded that the proposed distance models help to improve diversity in the population, but have no effect in the final solution performance for the experiments carried out. Hence the article contribution can be also assessed as a promising technique to improve performance of existing methods in time series modelling problems. However, this improvement in performance comes at the cost of a higher computational time required to calculate the distance among the members of the population.

## Acknowledgements

This work was supported by the projects PID2020-112495RB-C21 and B-TIC-42-UGR20.

## References

Affenzeller, M., Winkler, S. M., Burlacu, B., Kronberger, G., Kommenda, M., & Wagner, S. (2017). Dynamic observation of genotypic and phenotypic diversity for different symbolic regression gp variants. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion* GECCO '17 (pp. 1553–1558). ACM. doi:10.1145/3067695.3082530.

Alfaro-Cid, E., Esparcia-Alcázar, A., Sharman, K., d. Vega, F. F., & Merelo, J. J. (2008). Prune and plant: A new bloat control method for genetic programming. In *2008 Eighth International Conference on Hybrid Intelligent Systems* (pp. 31–35). doi:10.1109/HIS.2008.127.

Alonso, C. L., Puente, J., & Montaña, J. L. (2008). Straight line programs: A new linear genetic programming approach. In *20th IEEE International Conference on Tools with Artificial Intelligence* (pp. 517–524). volume 2. doi:10.1109/ICTAI.2008.14.

Angeline, P. J. (1994). Genetic programming and emergent intelligence. In *Advances in Genetic Programming* (pp. 75–98). MIT Press. doi:10.5555/185984.185992.

Aslam, M. W., Zhu, Z., & Nandi, A. K. (2018). Diverse partner selection with brood recombination in genetic programming. *Applied Soft Computing*, *67*, 558 – 566. doi:https://doi.org/10.1016/j.asoc.2018.03.035.

Badran, K., & Rockett, P. (2007). The roles of diversity preservation and mutation in preventing population collapse in multiobjective genetic programming. In *Proceedings of GECCO 2007: Genetic and Evolutionary Computation Conference* (pp. 1551–1558). doi:10.1145/1276958.1277272.

Billard, L., & Diday, E. (2002). Symbolic regression analysis. In *Classification, Clustering, and Data Analysis* (pp. 281–288). Berlin, Heidelberg. doi:10.1007/978-3-642-56181-8_31.

Brameier, M., & Banzhaf, W. (2001). Evolving teams of predictors with linear genetic programming. *Genetic Programming and Evolvable Machines*, *2*, 381–407. doi:10.1023/A:1012978805372.

Burke, E. K., Gustafson, S., & Kendall, G. (2004). Diversity in genetic programming: an analysis of measures and correlation with fitness. *IEEE Transactions on Evolutionary Computation*, *8*, 47–62. doi:10.1109/TEVC.2003.819263.

Burks, A. R., & Punch, W. F. (2017). An analysis of the genetic marker diversity algorithm for genetic programming. *Genetic Programming and Evolvable Machines*, *18*, 213–245. doi:10.1007/s10710-016-9281-9.

Burlacu, B., Kronberger, G., Kommenda, M., & Affenzeller, M. (2019). Parsimony measures in multi-objective genetic programming for symbolic regression. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion* GECCO '19 (pp. 338–339). ACM. doi:10.1145/3319619.3322087.

Castelli, M., Trujillo, L., Vanneschi, L., & Popovič, A. (2015). Prediction of energy performance of residential buildings: A genetic programming approach. *Energy and Buildings*, *102*, 67 – 74. doi:10.1016/j.enbuild.2015.05.013.

Chen, G., Low, C. P., & Yang, Z. (2009). Preserving and exploiting genetic diversity in evolutionary programming algorithms. *IEEE Transactions on Evolutionary Computation*, *13*, 661–673. doi:10.1109/TEVC.2008.2011742.

Claude, F., & Navarro, G. (2008). Indexing straight-line programs.

Cordón, O., Damas, S., & Santamaría, J. (2006). Feature-based image registration by means of the chc evolutionary algorithm. *Image and Vision Computing*, *24*, 525 – 533. doi:https://doi.org/10.1016/j.imavis.2006.02.002.

Ekárt, A., & Németh, S. Z. (2000). A metric for genetic programs and fitness sharing. In *Genetic Programming* (pp. 259–270). Berlin, Heidelberg. doi:10.1007/978-3-540-46239-2_19.

Ekárt, A., & Németh, S. Z. (2002). Maintaining the diversity of genetic programs. In *Genetic Programming* (pp. 162–171). Berlin, Heidelberg. doi:10.13140/RG.2.1.2876.0167.

Eshelman, L. J. (1991). The chc adaptive search algorithm: How to have safe search when engaging in nontraditional genetic recombination. *Foundations of Genetic Algorithms*, *1*, 265 – 283. doi:https://doi.org/10.1016/B978-0-08-050684-5.50020-3.

Eshelman, L. J., & Schaffer, J. D. (1993). Real-coded genetic algorithms and interval-schemata. In *Foundations of Genetic Algorithms* (pp. 187 – 202). Elsevier volume 2 of *Foundations of Genetic Algorithms*. doi:https://doi.org/10.1016/B978-0-08-094832-4.50018-0.

Ferdjoukh, A., Galinier, F., Bourreau, E., Chateau, A., & Nebut, C. (2017). Measuring differences to compare sets of models and improve diversity in mde.

Giles, J., Giles, J., & Society, A. M. (1987). *Introduction to the Analysis of Metric Spaces*. Australian Mathematical Society Lecture Series. Cambridge University Press.

Harrell, F. (2015). *Regression Modeling Strategies: With Applications to Linear Models, Logistic and Ordinal Regression, and Survival Analysis*. Springer Series in Statistics. Springer-Verlag New York. doi:10.1007/978-3-319-19425-7.

Hildebrandt, T., & Branke, J. (2015). On using surrogates with genetic programming. *Evolutionary Computation*, *23*, 343–367. doi:10.1162/EVCO_a_00133.

de Jong, E. D., Watson, R. A., & Pollack, J. B. (2001). Reducing bloat and promoting diversity using multi-objective methods. In *Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation* (pp. 11–18). doi:10.5555/2955239.2955241.

Kalkreuth, R., Rudolph, G., & Krone, J. (2015). Improving convergence in cartesian genetic programming using adaptive crossover, mutation and selection. In *2015 IEEE Symposium Series on Computational Intelligence* (pp. 1415–1422). doi:10.1109/SSCI.2015.201.

Kelly, J., Hemberg, E., & O'Reilly, U.-M. (2019). Improving genetic programming with novel exploration - exploitation control. In *Genetic Programming* (pp. 64–80). Springer International Publishing. doi:10.1007/978-3-030-16670-0_5.

Koza, J. R. (1992). *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge, MA, USA: MIT Press. doi:10.5555/138936.

Kulunchakov, A., & Strijov, V. (2017). Generation of simple structured information retrieval functions by genetic algorithm without stagnation. *Expert Systems with Applications*, *85*, 221 – 230. doi:https://doi.org/10.1016/j.eswa.2017.05.019.

Li, G., Wang, J., Lee, K., & Leung, K.-S. (2008). Instruction-matrix-based genetic programming. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, *38*, 1036–1049. doi:10.1109/TSMCB.2008.922054.

Li, Q., Cheng, H., & Yao, M. (2016). Adaptive multi-phenotype based gene expression programming algorithm. *Chinese Journal of Electronics*, *25*, 807–816. doi:10.1049/cje.2016.08.041.

Liu, L., Cai, H., Ying, M., & Le, J. (2007). Rlgp: An efficient method to avoid code bloating on genetic programming. In *2007 International Conference on Mechatronics and Automation* (pp. 2945–2950). doi:10.1109/ICMA.2007.4304028.

Lozano, M., Herrera, F., & Cano, J. R. (2008). Replacement strategies to preserve useful diversity in steady-state genetic algorithms. *Information Sciences*, *178*, 4421 – 4433. doi:https://doi.org/10.1016/j.ins.2008.07.031.

Lu, Q., Ren, J., & Wang, Z. (2016). Using genetic programming with prior formula knowledge to solve symbolic regression problem. *Intell. Neuroscience*, *1021378*, 1:1–1:1. doi:10.1155/2016/1021378.

Martín, D., Alcalá-Fdez, J., Rosete, A., & Herrera, F. (2016). Nicgar: A niching genetic algorithm to mine a diverse set of interesting quantitative association rules. *Information Sciences*, *355-356*, 208 – 228. doi:https://doi.org/10.1016/j.ins.2016.03.039.

Mc Ginley, B., Maher, J., O'Riordan, C., & Morgan, F. (2011). Maintaining healthy population diversity using adaptive crossover, mutation, and selection. *IEEE Transactions on Evolutionary Computation*, *15*, 692–714. doi:10.1109/TEVC.2010.2046173.

McKay, B., Willis, M. J., & Barton, G. W. (1995). Using a tree structured genetic algorithm to perform symbolic regression. In *First International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications* (pp. 487–492). doi:10.1049/cp:19951096.

Miller, J. F., & Thomson, P. (2000). Cartesian genetic programming. In *Genetic Programming* (pp. 121–132). Springer Berlin Heidelberg. doi:10.1007/978-3-540-46239-2_9.

Moraglio, A., Krawiec, K., & Johnson, C. G. (2012). Geometric semantic genetic programming. In *Parallel Problem Solving from Nature - PPSN XII*. Springer Berlin Heidelberg. doi:10.1007/978-3-642-32937-1_3.

Nicolau, M., Agapitos, A., O'Neill, M., & Brabazon, A. (2015). Guidelines for defining benchmark problems in genetic programming. In *2015 IEEE Congress on Evolutionary Computation (CEC)* (pp. 1152–1159). doi:10.1109/CEC.2015.7257019.

O'Neill, M., & Ryan, C. (2001). Grammatical evolution. *IEEE Transactions on Evolutionary Computation*, *5*, 349–358. doi:10.1109/4235.942529.

Pawlik, M., & Augsten, N. (2016). Tree edit distance: Robust and memory-efficient. *Information Systems*, *56*, 157 – 173. doi:https://doi.org/10.1016/j.is.2015.08.004.

dal Piccol Sotto, L. F., & de Melo, V. V. (2016). Studying bloat control and maintenance of effective code in linear genetic programming for symbolic regression. *Neurocomputing*, *180*, 79 – 93. doi:https://doi.org/10.1016/j.neucom.2015.10.109.

Qu, L., Hongbing, C., & Lin, H. X. (2015). Edit distance based crossover operator in gene expression programming. In *2015 8th International Conference on Biomedical Engineering and Informatics (BMEI)* (pp. 468–472). doi:10.1109/BMEI.2015.7401550.

Ristad, E. S., & Yianilos, P. N. (1998). Learning string-edit distance. *IEEE Transactions on Pattern Analysis and*

*Machine Intelligence*, *20*, 522–532. doi:10.1109/34.682181.

Rueda, R., Cuéllar, M., Pegalajar, M., & Delgado, M. (2019). Straight line programs for energy consumption modelling. *Applied Soft Computing*, *80*, 310 – 328. doi:https://doi.org/10.1016/j.asoc.2019.04.001.

Segura, C., Hernández-Aguirre, A., Luna, F., & Alba, E. (2017). Improving diversity in evolutionary algorithms: New best solutions for frequency assignment. *IEEE Transactions on Evolutionary Computation*, *21*, 539–553. doi:10.1109/TEVC.2016.2641477.

Smit, S. K., & Eiben, A. E. (2010). Using entropy for parameter analysis of evolutionary algorithms. In *Experimental Methods for the Analysis of Optimization Algorithms* (pp. 287–310). Berlin, Heidelberg: Springer. doi:10.1007/978-3-642-02538-9_12.

Ursem, R. K. (2002). Diversity-guided evolutionary algorithms. In *Parallel Problem Solving from Nature — PPSN VII* (pp. 462–471). Berlin, Heidelberg: Springer Berlin Heidelberg. doi:10.1007/3-540-45712-7_45.

Uy, N. Q., O'Neill, M., Hoai, N. X., Mckay, B., & Galván-López, E. (2010). Semantic similarity based crossover in gp: The case for real-valued function regression. In *Artifical Evolution* (pp. 170–181). Berlin, Heidelberg: Springer Berlin Heidelberg. doi:10.1007/978-3-642-14156-0_15.

Črepinšek, M., Liu, S.-H., & Mernik, M. (2013). Exploration and exploitation in evolutionary algorithms: A survey. *ACM Comput. Surv.*, *45*, 1–33. doi:10.1145/2480741.2480752.

Waterman, M., Smith, T., & Beyer, W. (1976). Some biological sequence metrics. *Advances in Mathematics*, *20*, 367 – 387. doi:10.1016/0001-8708(76)90202-4.