Unveiling the properties of structured grammatical evolution

Article *in* Genetic Programming and Evolvable Machines · September 2016

DOI: 10.1007/s10710-015-9262-4

CITATIONS
30

READS
158

3 authors:

Nuno Lourenço
University of Coimbra
65 PUBLICATIONS **275** CITATIONS
SEE PROFILE

Francisco B. Pereira
Instituto Politécnico de Coimbra
98 PUBLICATIONS **1,078** CITATIONS
SEE PROFILE

Ernesto Costa
University of Coimbra
216 PUBLICATIONS **1,955** CITATIONS
SEE PROFILE

Noname manuscript No.
(will be inserted by the editor)

Unveiling the Properties of Structured Grammatical Evolution

Nuno Lourenço · Francisco B. Pereira · Ernesto Costa

Received: date / Accepted: date

Abstract Structured Grammatical Evolution (SGE) is a new genotypic representation for Grammatical Evolution (GE). It comprises a hierarchical organization of the genes, where each locus is explicitly linked to a non-terminal of the grammar being used. This one-to-one correspondence ensures that the modification of a gene does not affect the derivation options of other non-terminals. We present a comprehensive set of optimization results obtained with problems from three different categories: symbolic regression, path finding, and predictive modeling. In most of the situations SGE outperforms standard GE, confirming the effectiveness of the new representation. To understand the reasons for SGE enhanced performance, we scrutinize its main features. We rely on a set of static measures to model the interactions between the representation and variation operators and assess how they influence the interplay between the genotype-phenotype spaces. The study reveals that the structured organization of SGE promotes an increased locality and is less redundant than standard GE, thus fostering an effective exploration of the search space.

Keywords Genetic Programming · Grammatical Evolution · Locality · Redundancy · Representation

N. Lourenço · F. B. Pereira · E. Costa
Centre for Informatics and Systems of the University of Coimbra, Polo II, Pinhal de Marrocos, 3030-290 Coimbra, Portugal
E-mail: naml@dei.uc.pt

F. B. Pereira
Polytechnic Institute of Coimbra, Quinta da Nora, 3030-199 Coimbra, Portugal
E-mail: xico@dei.uc.pt

E. Costa
E-mail: ernesto@dei.uc.pt

1 Introduction

Evolutionary Algorithms (EA) are computational methods inspired by the principles of natural selection and genetics. Over the years they have been successfully used in different situations, including optimization, design or learning problems. Genetic Programming (GP) is an EA branch that deals with the automatic evolution of computer programs/algorithmic strategies. One of the most relevant variants of GP is Grammatical Evolution (GE) [7,26,35], whose distinctive feature is how it decouples the genotype (a linear string) from the phenotype (a tree expression). GE relies on a mapping process to translate the linear string into an executable program. This transformation is guided by the production rules of a grammar that help to establish the set of syntactically correct programs.

GE has some known issues related with locality and redundancy [17,32, 36]. A representation is said redundant when several different genotypes correspond to one phenotype. O'Neill et al. [26] present some experimental results corroborating the benefits of redundancy. In another study, Rothlauf et al. revealed that, in about 90% of the applications, genotypic mutation does not modify the phenotype [31]. This is a highly debated topic among researchers, but, clearly, excessive redundancy levels slow down evolution, thus decreasing EA efficiency [31]. Locality studies how variations performed in the genotype reflect on the phenotype. An EA has high locality if small modifications on the genotype result in small modifications on the phenotype, thus creating conditions for an effective sampling of the search space. If this condition is not satisfied, the exploration performed by an EA tends to resemble random search [12,13,29,30]. The work presented in [32] reveals that GE has low locality as neighboring genotypes often do not correspond to neighboring phenotypes, thereby compromising a guided search around high quality solutions. In a recent paper, Whigham et al. [37] support this diagnosis and argue that GE standard mapping might not be an appropriate framework for grammar-based GP.

Structured Grammatical Evolution (SGE) is a novel genotypic representation for GE, where each gene is linked to a non-terminal of the grammar being used [20]. SGE ensures that the modification of a gene does not affect the derivation options of other non-terminals, thereby increasing locality. Moreover, the one-to-one correspondence between genes and non-terminals removes the need to rely on the modulo operation in the genotype-phenotype mapping, thus reducing the redundancy levels. The effectiveness of SGE was first assessed on a preliminary set of problems [20] and results were encouraging, as the new representation was able to obtain good optimization results. Also, it proved to be efficient, since it needed a lower number of evaluations to discover good quality solutions.

In this work we consider an extended set of benchmark optimization problems that confirm the effectiveness of SGE. In addition, we present an in-depth analysis that helps to gain insight into the distinctive features of SGE and to understand why it outperforms the standard GE representation in most of the

considered optimization scenarios. We rely on a set of static measures to characterize the interactions between the representation and variation operators and assess how they influence the interplay between the genotype-phenotype spaces. Specifically, we rely on the innovation measures proposed by Raidl et al. [29], which help to identify the locality and redundancy levels of the representations. The study clarifies how the different components of SGE help to reduce redundancy and increase locality, when compared to the standard GE representation.

We proceed by introducing GE and relevant related work in section 2. Then, in section 3 we provide a detailed explanation of SGE, focusing on the genotypic representation and the genotype-phenotype mapping. Section 4 details the results obtained by SGE and standard GE in several selected optimization problems. In section 5 we present a comparative analysis of the locality and redundancy levels of SGE and standard GE representations, whereas section 6 comprises a discussion of the main results obtained. Finally section 7 gathers the main conclusions and suggests some directions for future work.

2 Background

2.1 Grammatical Evolution

Grammatical Evolution (GE) is a form of Grammar-Based Genetic Programming (GBGP) [21] originally proposed by Ryan et al. [35]. As with standard GP, the goal of GE is to evolve executable algorithmic strategies. GE is different from other non grammar-based GP variants, for there is a clear separation between the linear genotypic string and the final phenotype, which is usually a program in the form of a tree expression. As a consequence, a mapping process is required to transform the string into an executable program, using the productions rules of a context-free grammar (CFG). A CFG is a tuple $G = (N, T, S, P)$, where N is a non-empty set of non-terminal symbols, T is a non-empty set of terminal symbols, S is an element of N called axiom, and P is a set of production rules of the form $A ::= \alpha$, with $A \in N$ and $\alpha \in (N \cup T)^*$. N and T are disjoint. Each grammar G defines a language $L(G)$ comprising all sequences of terminal symbols that can be derived from the axiom: $L(G) = \{w : S \overset{*}{\Rightarrow} w, w \in T^*\}$.

The translation of the genotype into the phenotype is done by simulating a leftmost derivation from the axiom of the grammar. This process scans the linear sequence from left to right and each integer (*i.e.*, each codon) is used to determine the grammar rule that expands the leftmost non-terminal symbol of the current partial derivation tree. Suppose that we have the following production rule identifying the four options to replace the axiom symbol $<expr>$:

$$
\begin{aligned}
<expr> ::= & <expr><op><expr> && (0)\\
| & (<expr>) && (1)\\
| & <pre-op>(<expr>) && (2)\\
| & <var> && (3)
\end{aligned}
$$

To select the alternative used to rewrite this symbol we take the first codon from the genotype, divide it by the number of options for $<expr>$ and take the remainder. Assuming that the first codon is 8, it follows that $8\%4 = 0$ and the axiom is rewritten as $<expr><op><expr>$. Then, this process is iteratively applied until the expression is composed only by terminals. Sometimes the length of the genotypic string is not sufficient to complete the mapping. In those cases the sequence is repeatedly reused in a process known as wrapping. If mapping exceeds a pre-determined number of wrappings, the process stops and the worst possible fitness value is assigned to the individual. For a detailed description of the process, consult [7,26].

2.2 Related Work

There are many reports in the literature addressing the extension/improvement of the original GE framework (see, *e.g.*, [17,22–24]). Additionally there are several studies that aim to gain insight on how GE explores the search space [32,36]. The contributions directly related with the current work are briefly reviewed in the next sections.

Mapping

The work of Keijzer et al. [17] is one of the first attempts to improve the mapping of standard GE by removing the bias that may exist when many rules from the grammar have the same number of production choices. Consider the following production set:

$$
\begin{aligned}
<bitstring> &::= <bit> | <bit><bitstring>\\
<bit> &::= 0 | 1
\end{aligned}
$$

In this example there are two possible choices for each rule. The modulo operation will always select the first option ($<bit>$ or 0) when an even codon is considered, whereas it will choose the second option ($<bit><bitstring>$ or 1) when odd codons are used. This creates a linkage bias between different productions, which may decrease GE effectiveness. To remove this effect they propose the *bucket rule*, a new mapping strategy that allows the same codon to select for different production choices.

The Chorus representation is loosely inspired on protein production to regulate the metabolic pathways of the cell and it pioneered the proposal of a position independent GE [2,34]. In this system the genotype is composed by 8-bit integers that represent the rules of the associated grammar. The key difference from GE is that each gene value corresponds to a specific production

rule. The modulo operation considers the total number of production rules in the grammar, so a specific allele always maps to the same rule, regardless of its position in the genotype. The analysis presented in the above mentioned reference reveals that Chorus obtains results comparable to standard GE in a set of selected optimization problems.

In [24], O'Neill et al. proposed πGE, another position independent alternative to GE. The standard GE mapping creates a positional dependency, as the derivation is always performed by expanding the leftmost non-terminal. πGE removes this bias by creating codons with two values: *nont* and *rule*. In this case, *nont* helps to select the next non-terminal NT to be expanded: $NT = nont\%count$, where *nont* is the value present in the genotype, and *count* is the number of non-terminals still in the derivation tree. The *rule* value of the codon pair, as in standard GE, selects which production rule should be applied from the selected non-terminal NT.

Fagan and coworkers [8,9] compared the performance of several mapping mechanisms. Besides the aforementioned πGE and the traditional depth-first expansion, they also considered breadth-first and a random expansion mechanism. Results revealed that πGE outperforms standard GE, confirming the relevance of investigating new, alternative, genotypic representations, and the corresponding mapping processes.

Representation and Search Operators

In 2002 Sullivan et al. [28] compared the performance of GE-based systems, while using different search strategies. They separately considered hill-climbing, simulated annealing, random search, and genetic algorithms as the GE search engine. Results obtained in several benchmark problems revealed that the genetic algorithm is the best option for achieving an enhanced performance. Additionally, the authors analysed the impact of different search operators and verified that crossover is critical for the success of the optimization. Alternative worth-mentioning search strategies embedded in GE-based systems include the application of differential evolution [22] or swarm algorithms [23].

Rothlauf et al. [32] pioneered the analysis of locality and redundancy in GE. Their research revealed that, in approximately 90% of the cases, a genotypic mutation does not change the phenotype. This result shows that GE suffers from extremely high levels of redundancy, which is mostly a consequence of the many-to-one mapping that allows multiple codons to correspond to the same production rule. Another important result of this work is related with the remaining 10% of mutations. Specifically, when the genotype suffers one mutation, changes of one or more units occur at the phenotypic level. Units of change in this level are estimated by the tree edit distance [39], that considers the minimal deletions, insertions, or replacements needed to transform one phenotype into the other. Empirical results obtained in two optimization problems reveal that the locality of the genotype-phenotype mapping is low, as many genotypic neighbors originate highly dissimilar phenotypes. Recently Thorhauer et al. [36] extended the previous work and compared the locality

of the standard GE operators with standard GP. They analyzed the locality on problems that rely on binary trees, by performing random walks through the search space and measuring the distance between parents and offspring. This research confirms the low locality of GE, reinforcing the relevance of developing alternative representations and variation operators.

Byrne et al. [4,5] proposed two distinct GE mutation operators, with complementary effects on the locality level: nodal mutation is a high-locality operator that changes the label of a single node in the derivation tree; structural mutation is a low-locality operator that modifies the structure of the derivation tree. An experimental study confirms the complementary effect of the two operators, as structural mutation promotes the exploration of the search space, whereas nodal mutation focuses on the exploitation of the neighborhood of current solutions.

In [6], the authors present an experimental study that identifies a positive correlation between the locus where variation operators are applied and their ability to change the phenotype. Results show that the application of crossover and mutation to the beginning of the genotype promotes low-locality, as it may alter the interpretation of all the remaining codons.

Hugosson et al. [15] analyze the impact of three different genotypic representations on GE effectiveness: binary, gray coding, and integer. Experimental results are inconclusive, as no representation clearly outperforms the others. According to the authors, this outcome suggests that the relevance of a given GE representation cannot be decoupled from the mapping process embedded in the grammar.

3 Structured Grammatical Evolution

Structured Grammatical Evolution (SGE) is a novel genotypic representation for Grammatical Evolution. In SGE each gene is linked to a specific non-terminal, and it comprises a list of integers used to select an expansion option. The length of each list is determined by computing the maximum possible number of expansions of the corresponding non-terminal (see details in sections 3.1 and 3.2). This structure ensures that the modification of a gene does not affect the derivation options of other non-terminals, thus limiting the number of changes that can occur at the phenotypic level. The values inside each list are bounded by the number of possible expansion options of the corresponding non-terminal. Therefore, mapping does not rely on the modulo rule, thus reducing the redundancy associated with it.

As an example, consider the grammar depicted in Fig. 1. The set of non-terminals is $\{<start>, <expr>, <term>, <op>\}$. Then, the SGE genotype is composed by four genes, each one linked to a specific non-terminal. To determine the length of the gene's lists we compute the maximum number of expansions of a non-terminal. The $<start>$ symbol is expanded only once, as it is the grammar axiom. The $<expr>$ symbol is expanded, at most, twice, because of the rule $<expr><op><expr>$. The computation of the list size for $<term>$

$<start> ::= <expr><op><expr>|<expr>$
$<expr> ::= <term><op><term>|(<term><op><term>)$
$<term> ::= x|0.5$
$<op> ::= +|-|*|/$

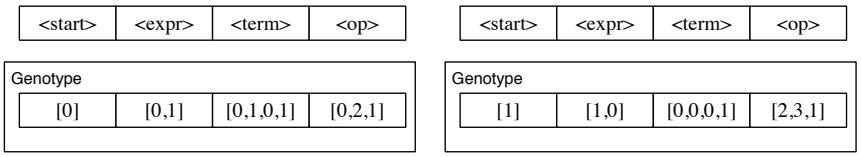
Fig. 1: Example of a production set to create polynomial expressions

establishes a direct dependence between this non-terminal and $<expr>$: each time $<expr>$ is expanded, $<term>$ is expanded twice (in the two possible expansion options). As the grammar allows a maximum of two $< expr >$ expansions, it immediately follows that the list size for the $<term>$ gene is four. Following the same line of reasoning, the list size for the $<op>$ gene is 3. Thus, the list sizes for each gene are: $<start> : 1, <expr> : 2, <term> : 4, <op> : 3$. To complete the list inside each gene we take the number of derivation options c_N of the corresponding non-terminal, and assign a random value from the interval $[0, c_N - 1]$ to every position. The $<start>$, $<expr>$ and $<term>$ symbols have $c_N = 2$, whereas $<op>$ has $c_N = 4$. Figs. 2a and 2b exemplify two possible SGE genotypes for this example.

The detailed derivation sequence of the genotype from Fig. 2a is presented in Fig. 3. Translation starts at the axiom ($<start>$) and it proceeds in the standard way by expanding non-terminals in a left-first manner. The first unused integer of the $<start>$ gene list is 0, which replaces the axiom with $<expr><op><expr>$. Given the 0 value in the first position of the $<expr>$ non-terminal, the expression is transformed into $<term><op><term><op><expr>$. Next, $<term>$ is replaced by x and the process continues until there are no more symbols to derive. The final derivation tree of the genotype from Fig. 2a (likewise Fig. 2b) is displayed in Fig. 2c (likewise Fig. 2d). The corresponding expressions are, respectively, $x + 0.5 \times (x - 0.5)$ and $(x \times x)$.

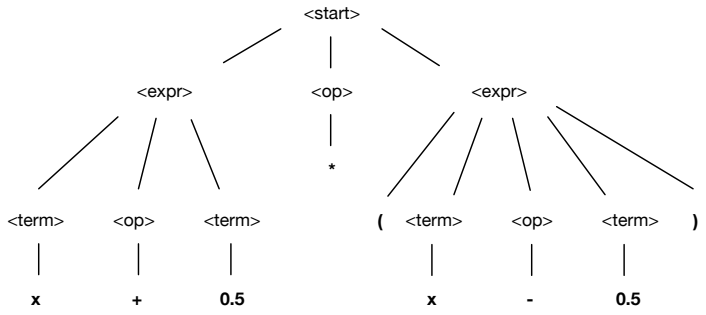
3.1 Genotypic Structure

To establish the structure of the genotype for a given grammar, one must compute an upper bound for the number of non-terminal expansions, as this defines the list size for each gene. This is accomplished in a two-step process. Initially, Alg. 1 iterates through the grammar productions and records the maximum number of non-terminal references that occur in each choice. At the same time, it builds a dictionary establishing a relation between non-terminals. In step 2, Alg. 2 iterates the set of non-terminals and recursively determines an upper bound for the number of expansions of each non-terminal. This algorithm takes into account the dependences between the non-terminals (e.g., the dependence between $<expr>$ and $<term>$ in the production set of Fig. 1).

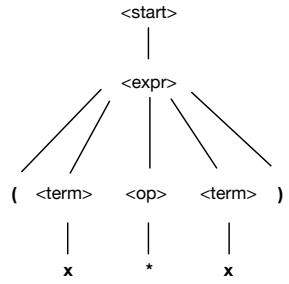


(a) SGE Genotype 1

(b) SGE Genotype 2



(c) Derivation Tree for Genotype 1



(d) Derivation Tree for Genotype 2

Fig. 2: Panels (a) and (b) exemplify two possible SGE genotypes for the production set of Fig. 1. Panels (c) and (d) display the corresponding derivation trees.

3.2 Recursive Grammars

The pre-processing described in the previous section does not consider recursive grammars. Standard GE deals with recursion by always trying to perform the translation into an executable program. If it runs out of integers, GE assigns the worst possible fitness value to the individual.

SGE deals with recursion in a different way, as it contains a parameter that establishes the maximum level of recursion. This value is selected prior to the

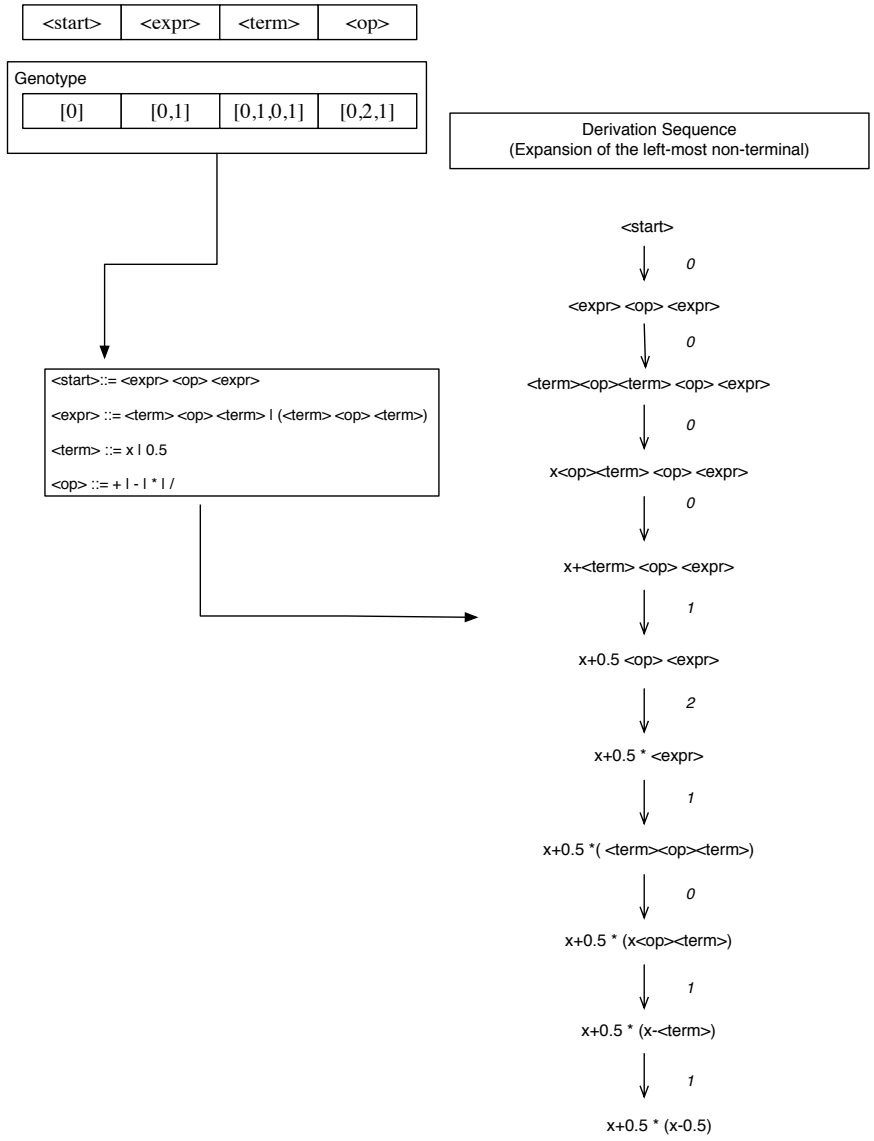


Fig. 3: Complete derivation sequence of the genotype presented in Fig. 2a.

application of the algorithm to a specific problem and, given this approach, a set of intermediate symbols that mimic the levels of the recursion tree must be inserted in the grammar. The following example is an excerpt of a grammar for symbolic regression problems:

Algorithm 1 Calculation of the non-terminal references.

```
countReferences ← {}
isReferencedBy ← {}
for nt in nonTerminalsSet do
    for production in grammar[nt] do
        for option in production do
            if option ∈ nonTerminalsSet then
                isReferencedBy[option] ← nt
                count[option] ← count[option] + 1
            end if
        end for
    end for
    for key in count do
        countReferences[key][nt] ← max(countReferences[key][nt], count[key])
    end for
end for
```

Algorithm 2 Calculation of the upper bound for non-terminals expansion.

```
function findReferences(nt, isReferencedBy, countReferencesByProd)
    r ← getTotalReferencesOfCurrentProduction(countReferencesByProd, nt)
    results ← []
    if nt = startSymbol then
        return 1
    end if
    for ref in isReferencedBy[nt] do
        result.append(findReferences(ref,isReferencedBy,countReferencesByProd))
    end for
    references ← references * max(result)

    return references
end function
```

$$
\begin{aligned}
&<start> ::= <expr>\\
&<expr> ::= <expr><op><expr> \mid <var>\\
&<op> ::= + \mid - \mid * \mid /\\
&<var> ::= x
\end{aligned}
$$

As the $<expr>$ production is recursive, it needs to be rewritten before the SGE structure is determined. Assuming that 2 levels of recursion were established it becomes:

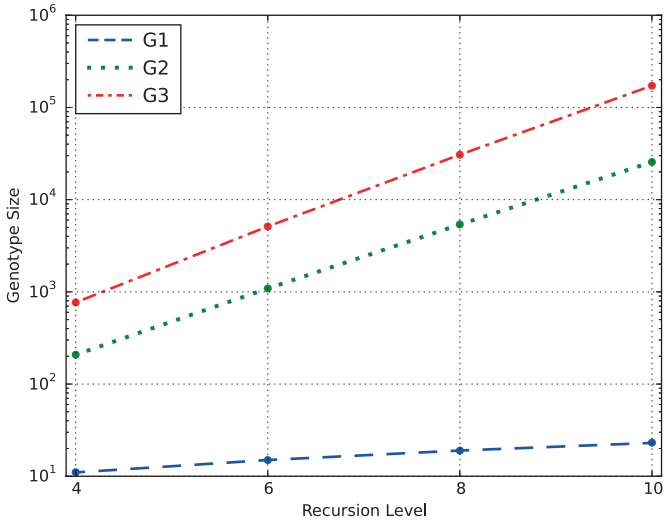


Fig. 4: SGE genotype size: four different levels of recursion and 3 grammars are considered (see grammar details in the text).

$$
\begin{aligned}
<start> &::= <expr> \\
<expr> &::= <expr\_lvl\_0><op><expr\_lvl\_0> \\
&\quad | <var> \\
<expr\_lvl\_0> &::= <expr\_lvl\_1><op><expr\_lvl\_1> \\
&\quad | <var> \\
<expr\_lvl\_1> &::= <var><op><var> | <var> \\
<op> &::= + \mid - \mid * \mid / \\
<var> &::= x
\end{aligned}
$$

The grammar transformation ensures that the original grammar symbols maintain their selection probability, since they are always replicated to each new added level. Also, there will be no invalid solutions. After unwrapping the recursive productions, Alg. 2 determines the upper bound for the number of expansions on each non-terminal. This ensures that genotype has enough values for decoding individuals and the mapping process will always end.

The grammar structure and the maximum recursion level impact the size of the SGE representation. Clearly, increasing the number of recursive productions, as well as increasing the maximum number of recursive calls per production, enlarges the structured organization of the genotype, thus directly impacting the size of the search space. To provide a visual illustrative example of the genotype growth we consider three simple grammars with dif-

ferent types of recursion: G1 is a grammar with one recursive production of the type <t> ::= <t>|λ; G2 references the same recursive production twice, <t> ::= <t><t>|λ; The third grammar G3 considers an additional recursive production that is referenced by another recursive production, i.e., <t> ::= <p>|<t><p> and <p> ::= <p><p>|λ. The results are depicted in Fig. 4. The horizontal axis considers four different levels of recursion, whereas the logarithmic vertical axis plots the size of the solution for the considered grammars. As expected, the genotype grows as the number of recursive calls increases. Also, it grows faster for grammars with chained recursive productions.

The current SGE proposal requires the specification of a fixed maximum recursion level. This establishes the size of the search space and constrains the structure of the evolved solutions by limiting the maximum program size. Future developments of the representation might improve on the fixed design, but, for now, users relying on SGE for optimization tasks should select a balanced limit that safely ensures the creation of solutions with the desired properties, without defining an excessively large search space. In Section 6 we present additional insight into the impact of this design option and reveal that limited variations in this setting do not affect search effectiveness. Finally, it is also worth noticing that most GP variants impose a constraint in the maximum program size, a mandatory step to prevent solutions from growing excessively and becoming computationally intractable. The constraint might be imposed in terms of tree depth, number of available nodes [19], or by imposing limits on the number of wrappings as performed in GE [7].

3.3 Genetic Operators

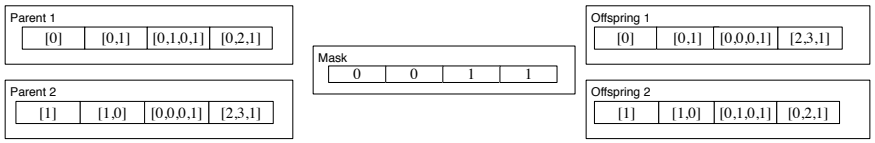
GE relies on standard operators to explore the search space, looking for promising solutions to the problem at hand. Two existing variation operators are adapted to work with SGE.

Recombination

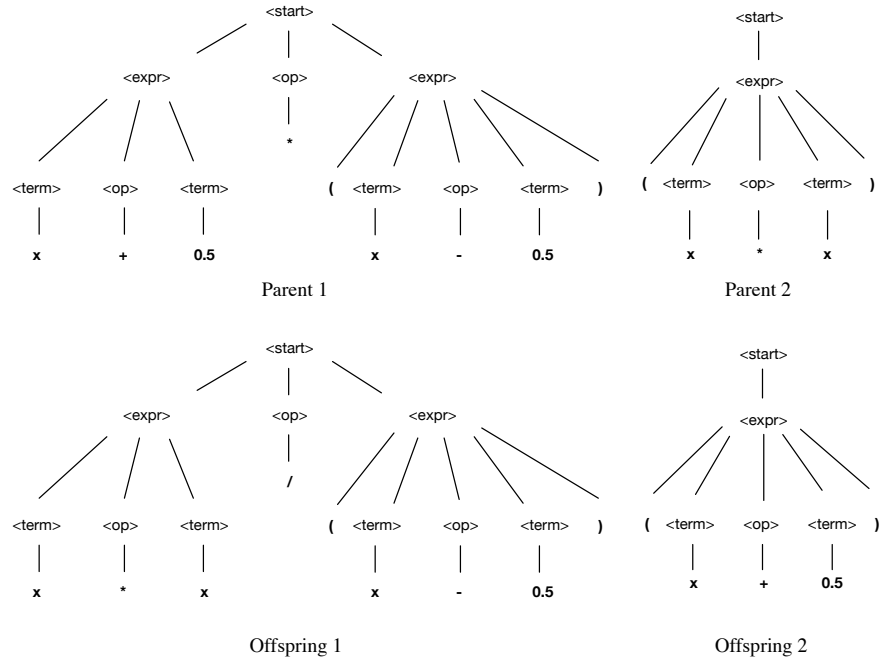
SGE recombination is based on the uniform crossover for binary representations. It generates a random binary mask and the offspring are created by selecting the parents genes' based on the mask values. Recombination does not modify the lists inside the genes. Fig. 5 illustrates an application of this operator.

Mutation

This operator has the ability to modify the lists inside the SGE genes. When applied to a specific list value, it changes it to a new random value from $[0, c_N - 1]$, where c_N is the number of derivations options of the non-terminal linked this gene. Fig. 6 illustrates an application of mutation.



(a) Crossover Application



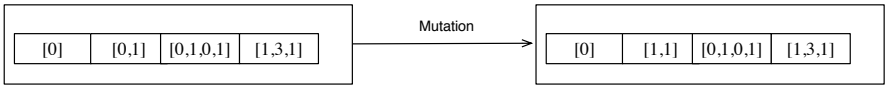
(b) Derivation Trees

Fig. 5: Application of the recombination operator. Panel (a) details the application at the genotypic level, whereas panel (b) illustrates changes in the corresponding derivation trees.

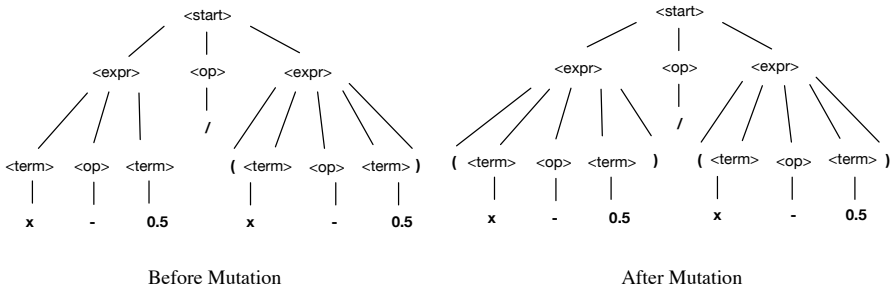
4 Experimental Results

This section presents a set of experimental results to assess the optimization performance of SGE and to compare it with standard GE. We selected several benchmark problems according to the guidelines proposed by White et al. [38]. These problems are classified in three categories: symbolic regression, path finding, and predictive modeling.

The GEVA [25] implementation of GE is adopted in the experiments. As SGE does not have invalid individuals, GE sensible initialization is used to create equivalent initial populations [33]. The complete experimental settings of both approaches are summarized in Table 1. The sum of the errors is the fitness function adopted for all tests and 30 independent runs were performed



(a) Mutation Application



(b) Derivation Trees

Fig. 6: Application of the mutation operator. Panel (a) details the application at the genotypic level, whereas panel (b) illustrates changes in the corresponding derivation trees.

Table 1: Settings for the Experimental Analysis

Parameter	GEVA	SGE
Initial Population	500	
Recombination rate	0.9	
Mutation rate	0.02	
Replacement	Steady-State with a generation gap of 0.9	
Selection	Tournament with size 3	
Generations	{50, 200}	
Recombination Operator	Single Point Crossover	SGE Crossover
Mutation Operator	Integer Flip Mutation	SGE Mutation
Genotype Size	128 (Ramped Half and Half Initialization)	-
Wraps	3	-
Max Level of Recursion	-	6

in each optimization scenario. Results presented are always averages of the 30 runs.

4.1 Symbolic Regression

Symbolic regression is perhaps the most well-known benchmark for GP. The goal is to approximate a function given a set of points and there are several polynomials that can be selected as target. In this work we consider two instances: the harmonic curve and the Pagie polynomial.

The main objective of the harmonic curve regression is to approximate the polynomial $\sum_i^x \frac{1}{i}$ in the interval $x \in [1, 50]$. In addition to the standard

optimization task, this problem also considers a generalization step that extrapolates the fitting to the interval $x \in [51, 120]$. The production set for the harmonic curve regression is defined as:

$$
\begin{aligned}
&<start> ::= <expr> \\
&<expr> ::= <expr><op><expr> \mid (<expr>) \\
&\qquad\quad \mid <pre\_op>(<expr>) \mid <var> \\
&<op> ::= + \mid * \\
&<pre\_op> ::= + \mid - \mid inverse \mid sqrt \\
&<var> ::= x
\end{aligned}
$$

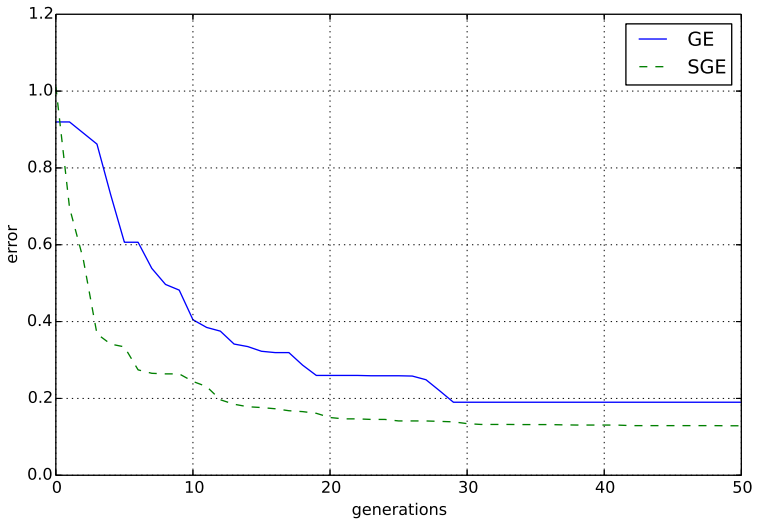
where $inverse$ is $1/x$.

In the Pagie polynomial, the goal is to approximate the function defined by $\frac{1}{1+x^{-4}} + \frac{1}{1+y^{-4}}$, where x, y are sampled in the interval $[-5, 5]$, with a step $s = 0.4$. Even though it defines a smooth search space, the Pagie polynomial has the reputation for being difficult [14,38]. As with the first task, this problem also considers a generalization step. Here, a compact grid of points, in which x, y are obtained from the same interval with a smaller step of $s = 0.1$, is defined. The production set for this problem is defined as:

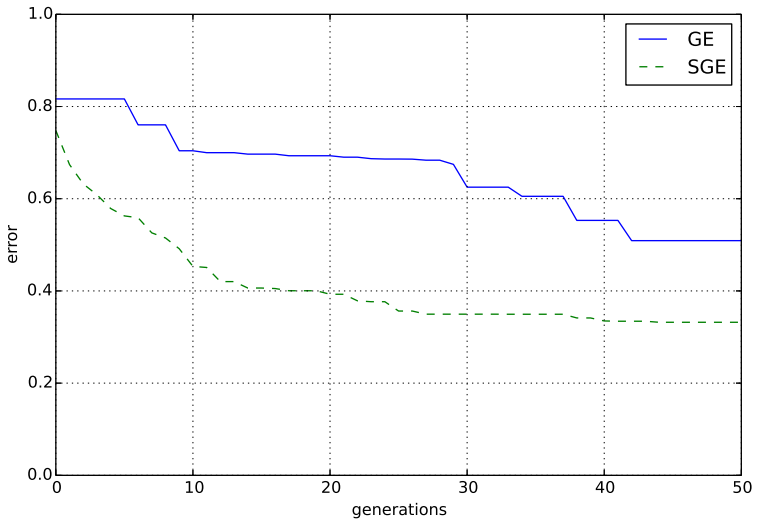
$$
\begin{aligned}
&<start> ::= <expr> \\
&<expr> ::= <expr><op><expr> \\
&\qquad\quad \mid (<expr>) \\
&\qquad\quad \mid <pre\_op>(<expr>) \\
&\qquad\quad \mid <var> \\
&<op> ::= + \mid - \mid * \mid / \\
&<pre\_op> ::= sin \mid cos \mid exp \mid log \\
&<var> ::= x \mid y
\end{aligned}
$$

The optimization results obtained by SGE and GE in the two regression problems are displayed in Fig. 7. The evolution of the Mean Best Fitness (MBF) confirms that both approaches gradually discover better approximations for the polynomials under consideration. There are, however, important differences between GE and SGE. In both problems, the results obtained by SGE are consistently better than those of GE throughout the optimization run, revealing that the new representation is more effective in building models to accurately approximate the target polynomial in the given interval. Additionally, SGE can deliver good quality solutions faster than standard GE, revealing an enhanced efficiency.

To complement the analysis we verified how some evolved models generalize to the extended intervals. For each representation, we selected the best solutions from the initial population (GE1, SGE1), from the population of the 25th generation (GE25, SGE25) and from the last population (GE50, SGE50). The results of the application of these models to the generalization step are presented in the boxplots of Fig. 8. For the harmonic curve, the general trend is for an enhanced performance of models discovered in the last generations.

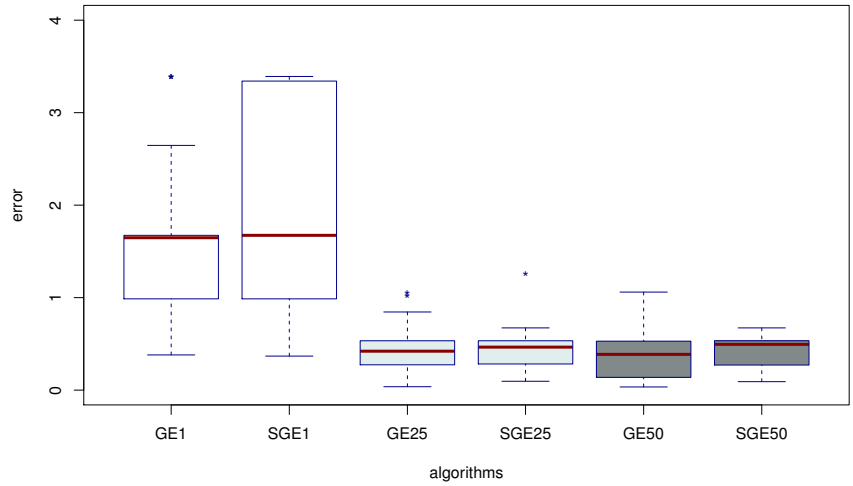


(a) Harmonic Curve

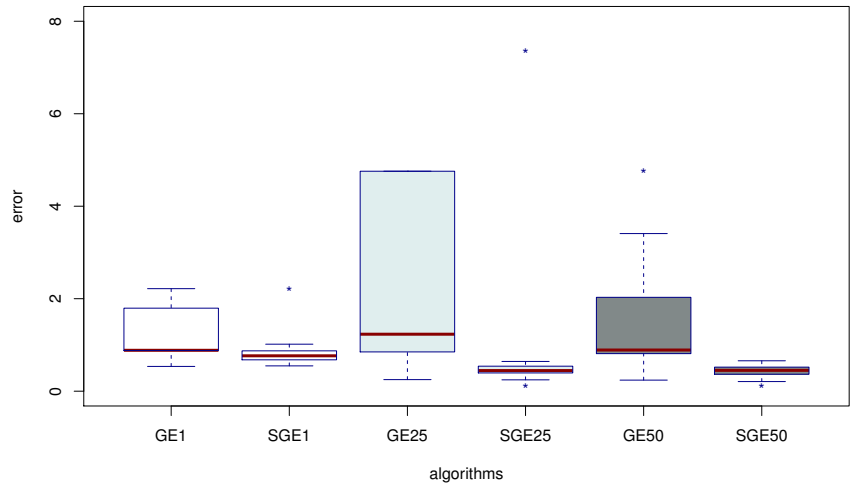


(b) Pagie

Fig. 7: Evolution of the MBF in the two regression problems: (a) Harmonic Curve; (b) Pagie.



(a) Harmonic Curve



(b) Pagie

Fig. 8: Results obtained by selected evolved models in the generalization step of the regression problems: (a) Harmonic Curve; (b) Pagie.

This is a relevant result, as it reveals that neither GE or SGE are overfitting. A closer inspection reveals that both GE variants obtain comparable results, and there are never statistically significant differences between pairs of models taken from the same generation. This outcome suggests that, in this particular problem, SGE and GE have similar generalization ability. The scenario on the Pagie polynomial generalization task is different. In this problem, SGE is able to clearly find strategies that generalize better than standard GE. Also, SGE variance is smaller, revealing an increased reliability. A direct comparison of the GE boxes in the generalization task discloses an overfitting situation, since the errors increase as we consider models obtained in later generations.

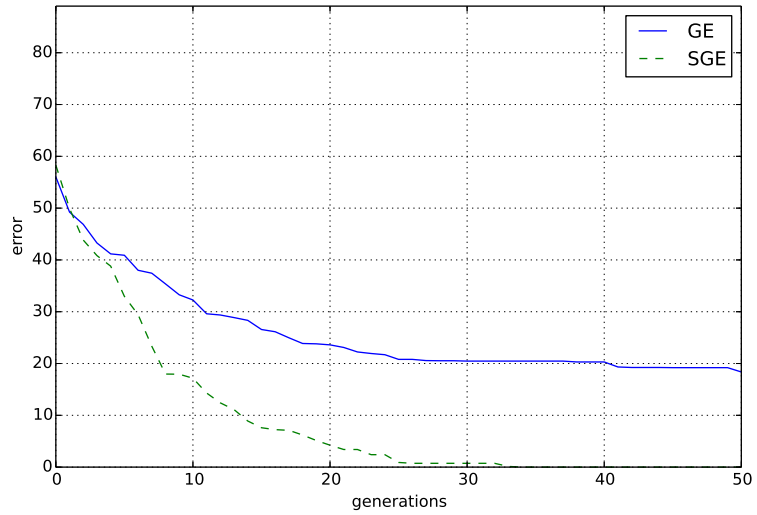
4.2 Path Finding

The goal of path finding problems is to evolve a set of instructions for an artificial agent so that it can collect a certain number of food pellets in a limited number of steps. In our study we consider two well-know trail instances: The Santa Fe and the Los Altos Hills. The first is an historical GP instance consisting of 89 food pellets distributed in a 32x32 toroidal grid, whereas the Los Altos Hills comprises 157 food pellets distributed in a 100x100 toroidal grid. In both problems the agent starts in the top-left corner of the grid and can turn left, right, move one square forward, and check if the square ahead contains food. The production set for these problems is:

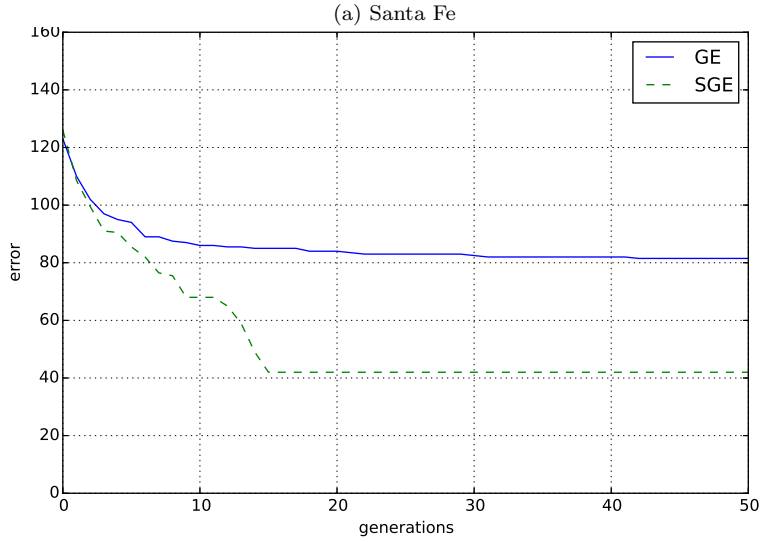
$$
\begin{aligned}
&\langle start\rangle ::= \langle code\rangle\\
&\langle code\rangle ::= \langle line\rangle\\
&\qquad\qquad |\langle code\rangle\\
&\qquad\qquad \langle line\rangle\\
&\langle line\rangle ::= if\ ant.sense\_food() :\\
&\qquad\qquad\quad \langle line\rangle\\
&\qquad\qquad else :\\
&\qquad\qquad\quad \langle line\rangle\\
&\qquad\qquad |\langle op\rangle\\
&\langle op\rangle ::= ant.turn\_left()\\
&\qquad\qquad |ant.turn\_right()\\
&\qquad\qquad |ant.move\_forward()
\end{aligned}
$$

The results obtained with both trail instances are presented in Fig 9. They follow the same trend and reveal that SGE is more effective than GE in the task of evolving strategies to collect the food pellets from the environment. The Santa Fe instance was included mostly for historical reasons, despite all recent debate concerning its relevance as a GP benchmark [38]. This problem was easily solved by SGE, as it quickly discovered a strategy to collect all food pellets. On the opposite, GE failed to discover optimal strategies for the agent. As for the Los Altos Hills, both representations converge to solutions that are unable to completely solve the problem. However, the strategies evolved by

SGE are clearly more effective in the task of discovering and collecting food pallets.



(a) Santa Fe



(b) Los Altos Hills

Fig. 9: Evolution of the MBF in the two path finding problems: (a) Santa Fe; (b) Los Altos Hills.

4.3 Predictive Modeling

Predictive modeling considers sets of previously labelled data to create models that correctly predict the label of unseen data. For our experiments we selected Bio, PPB and LD50, three high-dimensional datasets from the pharmacokinetics domain. In the Bio dataset the goal is to predict the human oral bioavailability (represented as %F). %F is the parameter that measures the percentage of the initial orally submitted drug dose that effectively reaches the systemic blood circulation after passing through the liver. This dataset consists of 359 instances of 242 elements (241 descriptors that identify a drug, followed by the %F value for that drug). The aim of the PPB dataset is to predict the protein-plasma binding level (%PPB), *i.e.*, to quantify the initial dose of a drug that reaches the blood circulation and binds to the plasma proteins. PPB is composed by 131 instances of 627 descriptors (626 features that identify a drug, followed by the known %PPB value for that drug). Lastly, the goal of LD50 is to build a model that predicts the median lethal dose of a set of drug compounds. This dataset is composed by 234 instances where each is a vector of of 627 elements (626 features that identify a drug, followed by the known LD50 value for that drug). For a more detailed description of the datasets please refer to [1,11].

The experiments described in this section correspond to a typical machine learning task and the datasets are divided in two equal parts. 50% of the instances are used in training, which is when the GE variants try to evolve promising models[1]. Afterwards, the remaining 50% are used to assess the generalization ability of the best evolved models. Following the guidelines of the aforementioned references, the number of generations of the evolutionary process is increased to 200 generations. The production set for these problems is:

$$
\begin{aligned}
&\langle start\rangle ::= \langle expr\rangle \\
&\langle expr\rangle ::= \langle expr\rangle\langle op\rangle\langle expr\rangle \mid (\langle expr\rangle) \\
&\qquad\quad \mid \langle var\rangle \\
&\langle op\rangle ::= + \mid - \mid * \mid / \\
&\langle var\rangle ::= x_i \mid -1.0 \mid -0.5 \mid 0.0 \mid 0.5 \mid 1.0
\end{aligned}
$$

Figs. 10a), 11a), 12a) present the evolution of the MBF during the training period, both for GE and SGE. The outcomes are in line with the optimization results obtained in previous problems and confirm SGE effectiveness and efficiency: the new representation is able to discover better solutions in a lower number of generations.

Once again we selected three models evolved with each GE variant to estimate the generalization ability: one from the beginning, one halfway and another at the end of the optimization. The results are depicted in the box-

[1] In the previous problems, optimization designates the step where the GE algorithms are applied. For predictive modeling tasks we adopt the standard terms from the ML community: training/generalization

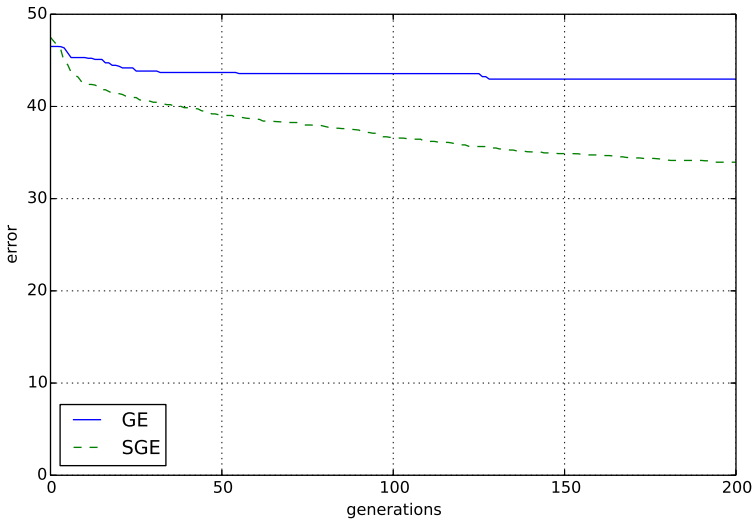
plots from Figs. 10b), 11b),12b). An inspection of results reveals two different scenarios: overfitting does not occur for the Bio and PPB datasets, as models from the final training stages generalize better. Furthermore, a direct comparison between GE200 and SGE200 demonstrates a clear advantage of the solutions evolved with SGE. In both datasets, the new representation promotes the discovery of models with increased reliability for labelling unseen data.

The situation for the LD50 dataset (Fig. 12b)) is different. The generalization behavior of models evolved with the two representations is similar, as the boxplots are leveled. Also, the solutions obtained in the last training stages are not clearly better than those discovered in the beginning, suggesting that the GE variants are not able to evolve reliable and general models. The LD50 dataset is considered as particularly challenging and standard GP-based approaches are unable to learn models with good generalization ability [11]. The solution to overcome difficulties is to rely on specific training strategies that help to control overfitting. Since our experiments did not consider such methods, it is not surprising that both GE variants failed to evolve effective models and obtain comparable generalization results.

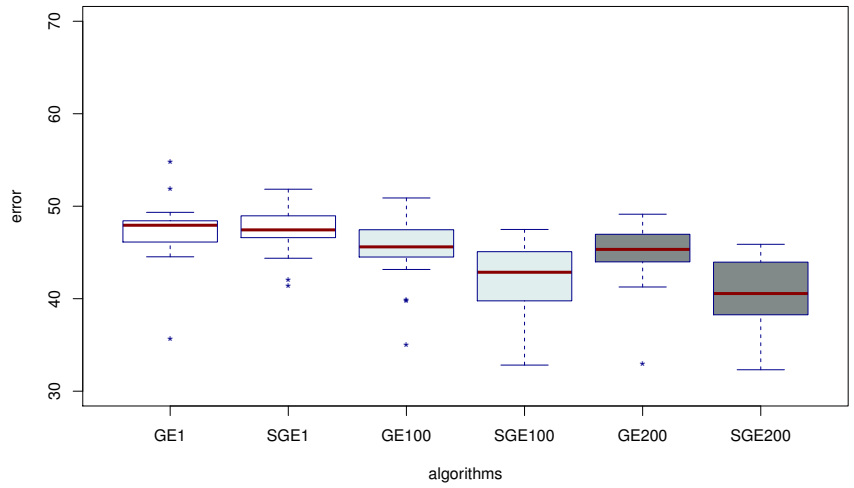
4.4 Statistical Validation

A statistical analysis was applied to confirm the relevance of the results. It estimates if there are differences in the means and, if that is the case, how significant they are. Since the samples do not follow a normal distribution, the analysis is performed using non-parametric tests. Moreover, and since the two groups are unrelated, the Mann-Whitney test with a significance level $\alpha = 0.05$ is adopted. When significant differences exist, the effect size r estimates its magnitude [10]: a $+++$ sign indicates that the effect size is large ($r >= 0.5$), a $++$ sign indicates that the effect size is medium ($0.3 <= r < 0.5$), whereas a $+$ identifies a small effect size ($0.1 <= r < 0.3$). Additionally, $\sim$ is used to identify situations where no statistical differences exist.

Table 2 reports the statistical analysis for the optimization/training stages of the 7 selected problems. Columns *GE* and *SGE* display, respectively, the MBF and standard deviation of the solutions from the final generation. The last column (*StatisticalValidation*) shows the *p-values* obtained in the comparison, and the corresponding effect size. Table 3 compares the generalization ability of the best models obtained in the last generation of optimization/training. The path finding problems are not included, as they do not have a generalization step. The results contained in both tables confirm the effectiveness of SGE. In the optimization/training stage, the new representation always finds solutions that significantly outperform standard GE. Even more important is the analysis summarized in Table 3, as it shows that the optimization effectiveness does not compromise the generalization ability. In three of the problems, SGE is able to evolve models that outperform standard GE, whereas in the remaining two the behavior of both representations is equivalent.

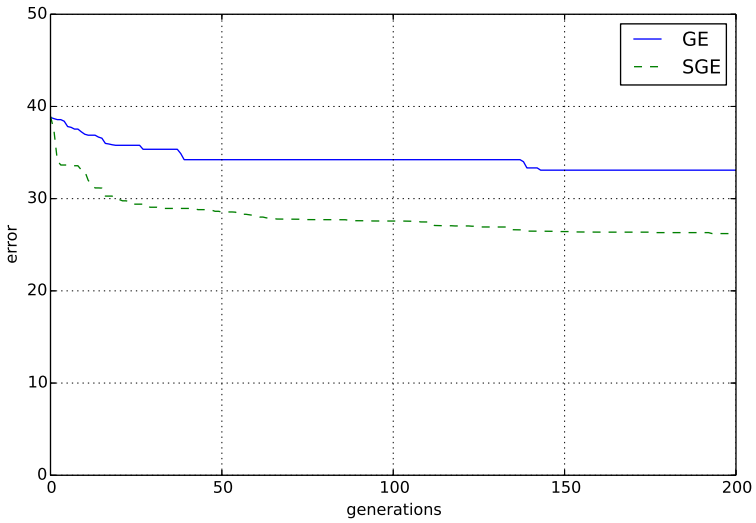


(a) Training

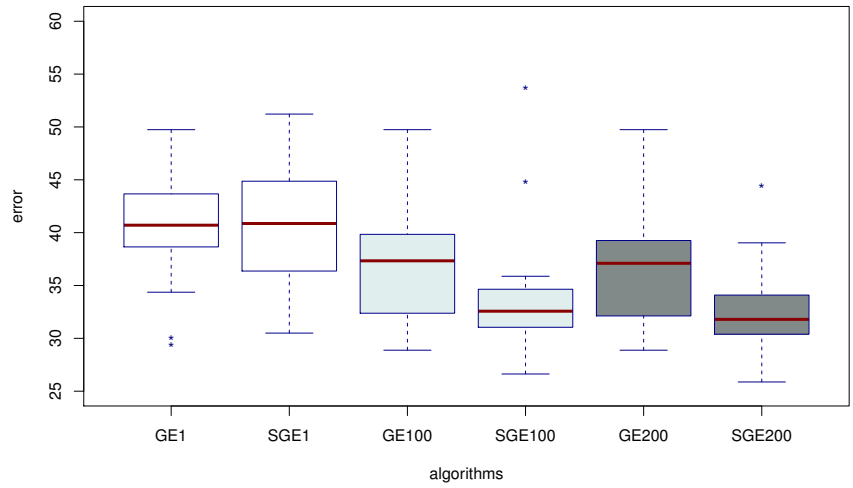


(b) Generalization

Fig. 10: Results obtained with the Bio dataset: (a) Evolution of the MBF during training; (b) Generalization ability of selected evolved models.

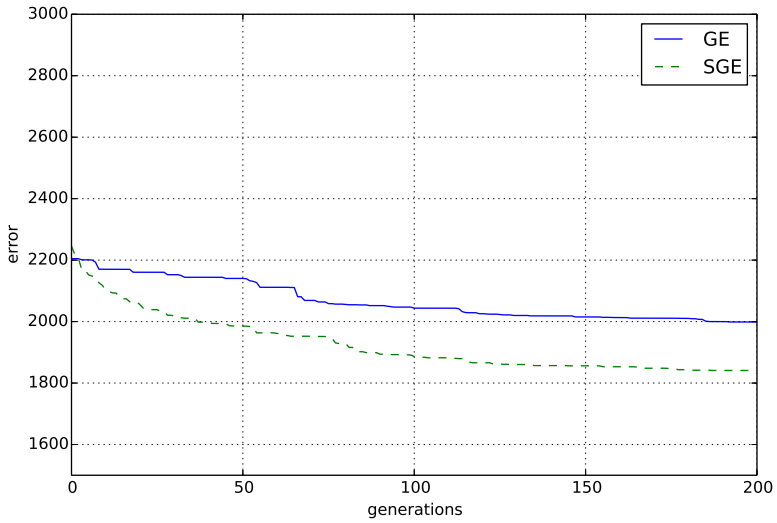


(a) Training

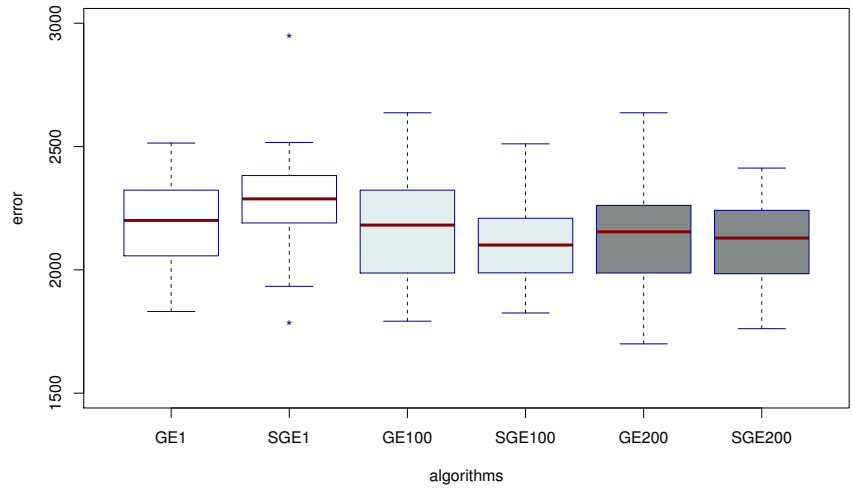


(b) Generalization

Fig. 11: Results obtained with the PPB dataset: (a) Evolution of the MBF during training; (b) Generalization ability of selected evolved models.



(a) Training



(b) Generalization

Fig. 12: Results obtained with the LD50 dataset: (a) Evolution of the MBF during training; (b) Generalization ability of selected evolved models.

Table 2: Statistical analysis of the Optimization/Training results for the 7 selected problems. Columns GE and SGE present the MBF and standard deviation, whereas the last column details the statistical validation. Results are averages of 30 runs.

Problem	GE	SGE	Statistical Validation	
			p-value	Effect Size
Harmonic	0.20 (±0.11)	0.13(±0.05)	$7.21 * 10^{-03}$	++
Pagie	0.50 (±0.26)	0.29 (±0.09)	$2.20 * 10^{-06}$	+++
Santa Fe	21.40 (±12.40)	0.00 (±0.00)	$9.21 * 10^{-14}$	+++
Los Altos Hills	81.90 (±1.90)	53.60 (±18.34)	$2.42 * 10^{-06}$	+++
BIO	42.97 (±4.95)	34.43 (±3.57)	$4.03 * 10^{-09}$	+++
PPB	34.06 (±5.06)	26.61 (±2.70)	$8.02 * 10^{-10}$	+++
LD50	2115.70 (±143.97)	1841.69 (±160.33)	$1.37 * 10^{-08}$	+++

Table 3: Statistical analysis of the Generalization step for the best models obtained in the last generation of optimization/training. Columns GE and SGE present the MBF and standard deviation, whereas the last column details the statistical validation. Results are averages of 30 runs.

Problem	GE	SGE	Statistical Validation	
			p-value	Effect Size
Harmonic	0.45 (±0.36)	0.56(±0.24)	0.05	∼
Pagie	0.90 (±6.58)	0.4352 (±0.13)	$3.02 * 10^{-09}$	+++
BIO	45.05 (±3.04)	40.67 (±3.84)	$2.44 * 10^{-06}$	+++
PPB	36.87 (±5.33)	32.35 (±3.70)	$1.96 * 10^{-04}$	+++
LD50	2148.00 (±202.17)	2162 (±352.49)	0.69	∼

5 Representation Analysis

There are several empirical tools that help to gain insight into the locality and redundancy of EAs. One of the first proposals was the Fitness Distance Correlation (FDC) [16]. FDC estimates the relation between fitness values and the distance to the optimum. If the quality increases as the distance the optimum decreases, then EAs are expected to have a good performance. However, FDC has some limitations, such as the requirement to know the global optima beforehand, which is impossible for many situations.

In this work we adopt another framework, originally proposed by Raidl et al. [29]. This empirical model allows for an easy and accurate analysis of the interplay between representation and genetic operators and provides a reliable basis for assessing the performance of the search algorithm. Studies performed with this framework are based on distance measures applied to pairs of solutions linked by the application of variation operators. The distribution of the distance values helps to estimate the redundancy of the representation and to establish a relation between the exploration of genotypic space and how it reflects in the phenotypic space. In concrete, the framework relies on Mutation

Innovation (MI) and Crossover Innovation (CI) measures. MI estimates how the mutation operator modifies the phenotype of an individual, whilst the CI approximates the novelty introduced by the recombination operator.

5.1 Basic Concepts

Spaces

GE relies on a mapping between genotypes and phenotypes, thus requiring the explicit definition of the two spaces: the genotype space ϕ_g, and the phenotype space ϕ_p. The genetic operators are applied to solutions x, on ϕ_g. Each x is mapped to a program X, in the phenotype space, via a mapping function f_g, such that $f_g(x) \rightarrow X$. Finally, the quality of each program X is measured by a fitness function f, on ϕ_p: $f(X) : \phi_p \rightarrow \mathbb{R}$.

Distances

The genotypic distance, d^g, is the Hamming distance for integer representations, *i.e.*, the distance between two arbitrary solutions $x, y \in \phi_g$ is defined as the total number of positions in which they differ.

In GE the phenotypes are derivation trees, allowing the application of the edit distance to measure the phenotypic distance, d^p, between two programs. The edit distance calculates the minimum number of elemental operations required to transform one tree into the other. There are three elemental operations:

1. **Deletion**: A node is removed from the tree;
2. **Insertion**: A node is added to the tree;
3. **Replacement**: The label of a node is modified.

All operations have unitary cost. The tree edit measure has been widely used to estimate the similarity between trees in GP [3,18,27,32].

Mutation Innovation

Let x be a solution in the genotype space and x^m the solution that results from the application of one mutation to x. Let $X, X^m \in \phi_p$ be the programs mapped by x and x^m, respectively. MI is the phenotypic distance between programs X and X^m:

$$MI = d^p(X, X^m) \tag{1}$$

The distribution of the MI variable discloses several important features concerning locality and redundancy. The application of a locally strong operator implies that a small modification in the genotype (such as the one originated by one mutation), should result in a limited phenotypic change, *i.e.,* the edit distance between the two involved programs should be small. On

the contrary, mutation operators with weak locality induce large phenotypic jumps, compromising search space exploitation.

When $MI = 0$, the mutation was not able to modify the phenotype of an individual. The frequency of these events helps to gain insight into the redundancy level of the representation.

Crossover Innovation

Crossover plays an important role in mixing up relevant blocks of solutions, thereby fostering the appearance of novel strategies. When using crossover, an offspring x^c is created from two parent solutions x^{p1}, x^{p2} (without loss of generality, in our analysis we disregard the second solution that results from the application of crossover). Let X^c, X^{p1} and $X^{p2} \in \phi_p$ be the corresponding phenotypes. CI measures the phenotypic distance between an offspring and its phenotypically closer parent:

$$CI = min(d^p(X^c, X^{p1}), d^p(X^c, X^{p2})) \tag{2}$$

If $CI = 0$, then $X^c = X^{p1}$ or $X^c = X^{p2}$, indicating that crossover was not able to create a phenotypically different solution. Moreover, it is expected that CI is directly related to the distance $d^g(x^{p1}, x^{p2})$ between parents, *i.e.*, dissimilar parents should increase the likelihood of generating innovative solutions.

5.2 Results

This section presents a comparative study between SGE and standard GE, to better understand why SGE exhibits an enhanced performance. We rely on the previously defined MI and CI measures to estimate the locality and redundancy levels of both representations and consider a recursive grammar regularly used for symbolic regression problems (Fig. 13). To compute the phenotypic distances, we adopt the dynamic programming approach proposed by Zhang et al. [39].

A few settings must be defined for each representation: standard GE requires 128 codons and the maximum number of wraps is set to 3, whereas the maximum recursion level of SGE is 6. These are the same settings that were adopted in the experiments presented in Section 4 and grant similar conditions to both representations, as they allow the generation of complete derivation trees with depth 6.

Redundancy

To estimate redundancy we count how many genotypic variations result in $MI = 0$ or $CI = 0$, *i.e.*, do not lead to phenotypic modifications. The first set of experiments considers the application of one mutation to a randomly generated solution. This process was repeated 10000 times. Results reveal that, for GE, approximately 90.3% of the mutations do not lead to a phenotypic

$$N = \{expr, op, pre\_op, var\}$$
$$T = \{sin, cos, exp, log, +, -, *, /, x, 1.0, (,)\}$$
$$S = \{start\}$$

And the production set P is:

$$\begin{aligned}
<start> &::= <expr> \\
<expr> &::= <expr><op><expr> \\
&\quad |(<expr>) \\
&\quad |<pre_op>(<expr>) \\
&\quad |<var> \\
<op> &::= + \mid - \mid * \mid / \\
<pre_op> &::= sin|cos|exp|log \\
<var> &::= x1|1.0
\end{aligned}$$

Fig. 13: Symbolic regression grammar.

change, whereas in SGE this percentage drops to about 40%. The GE redundancy is in line with the values presented in [32] and the disparity between the two representations is explained by the different mapping strategies: SGE eliminates all redundancy that results from the modulo rule, as there is a direct relation between the values inside the genes' lists and the alternative derivation options of the corresponding non-terminal.

To complement the analysis, we also measured the redundancy level after a sequence of mutations is applied to a solution. In concrete, we consider a random walk where we depart from a random solution and iteratively apply a mutation event. At each step, the phenotypic distance between the original solution and the current mutant is calculated. Results presented are averages of 10000 random walks with $k = 20$ steps. To simplify the analysis, the phenotypic distances between the original solution and the successive mutants are grouped in different sets. Given a d^p distance between two solutions, the set D_i to which it is assigned, is determined in the following way: $\{D0 : d^p = 0; D1 : d^p = 1; D2 : 1 < d^p \leq 5; D3 : 5 < d^p \leq 10; D4 : 10 < d^p \leq 20; D5 : 20 < d^p \leq 30; D6 : 30 < d^p \leq 40; D7 : 40 < d^p \leq 50; D8 : 50 < d^p \leq 60; D9 : 60 < d^p \leq 70; D10 : 70 < d^p \leq 80; D11 : 80 < d^p \leq 90; D12 : 90 < d^p \leq 100; D13 : d^p \geq 100; \}$.

The boundaries selected for the intervals were adjusted in such a way that they help to emphasize the distribution of distances. The two panels of Fig. 14 contain the distribution of distances (vertical axis) over the 20 steps of the random walk (horizontal axis): panel (a) displays results obtained with standard GE, while panel (b) shows the outcomes of SGE. For each mutation step k, the gray shaded square represents the percentage of k-mutated individuals, whose phenotypic distance to the original solutions falls in that specific dis-

tance set. The darker the squares, the higher the percentage of individuals in that situation.

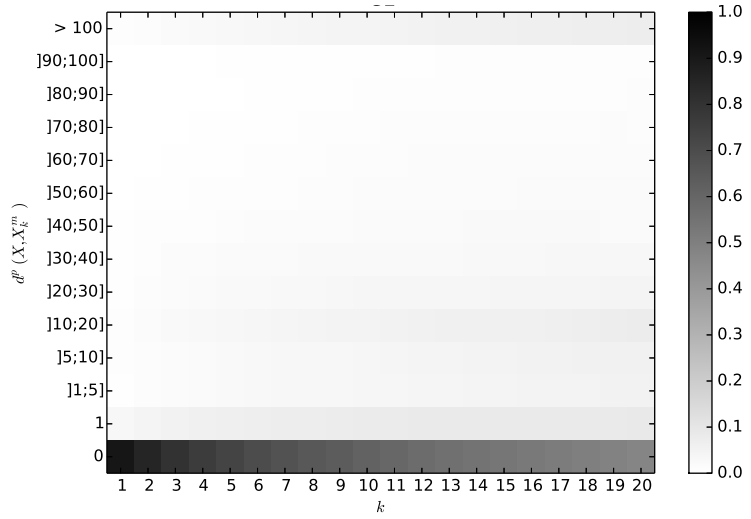
For GE, the application of a single mutation leads to a situation where about 90.3% of the individuals belong to the set $D0$, 4.5% belong to $D1$, 2.6% belong $D2$, and 1.1% belongs to $D3$, and so on. Results from panel (a) reveal that GE redundancy decreases, as mutations gradually start to accumulate. Nevertheless, nearly half of the random walks end in solutions whose phenotypic distance to the starting point is still 0, *i.e.*, 20 mutations were not enough to obtain a different derivation tree. It is obvious that the absolute values are dependent on several design options (e.g., the number of codons in the genotype). However, these results confirm that standard GE is vulnerable to extremely high level of redundancy, which clearly compromises search efficiency.

The evolution of the SGE redundancy levels is displayed in panel (b) of Fig. 14. The general trend is more in accordance to what is expected when mutations start to accumulate. After one step, around 40% of mutants remain identical to the original solutions, but redundancy gradually drops as successive mutations are applied and, when $k = 20$, the percentage is less then 10%. Moreover, the plot shows that the phenotypic distances are scattered across the defined sets, suggesting that mutation allows SGE to perform a meaningful exploration of the search space.

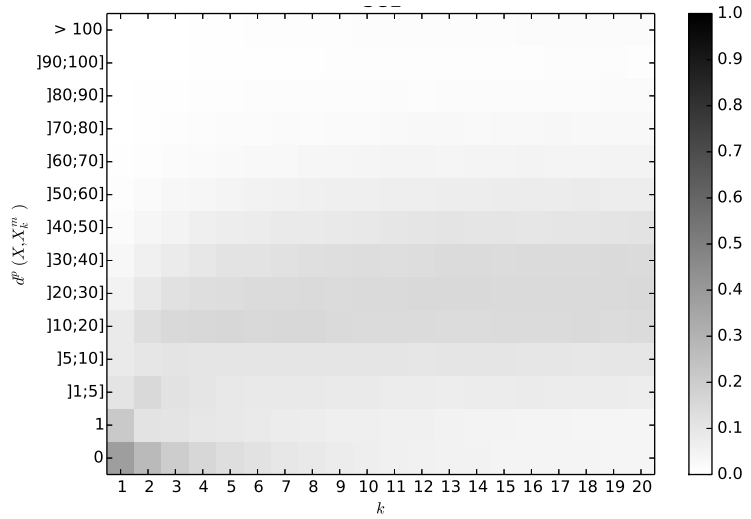
The CI measure helps to estimate how crossover impacts redundancy. In this experiment, one parent x^{p1} is randomly generated and kept unchanged throughout all the test. The second parent x^{p2} is obtained from the first by the application of $k > 0$ successive mutations. At each step, crossover is performed, one of the offspring is randomly chosen, and the corresponding CI is measured. The process ends after 20 steps. All results presented are averages of 10000 runs. Fig. 15 displays the probability of creating a phenotypically identical offspring ($CI = 0$), as the genotypic distance between parents increases (horizontal axis). For both representations, $P(CI_k = 0)$ naturally decreases with increasing k. However, for all k values, SGE clearly achieves higher innovation levels, thus fostering the discovery of novel solutions. Besides the existence of the modulo rule, another plausible explanation for the lower innovation of standard GE is related to the, possibly large, non-expressed area of its genotype. When the cut point falls in this area, the expressed phenotype will not change and the crossover operation will have no immediate impact in the discovery of novel solutions.

Locality

We will now focus in the subset of situations where $d^p > 0$, *i.e.*, where the application of variation operators originates a different phenotype. In Fig. 16 we present the phenotypic distances between an original random solution and mutants iteratively generated. Results are averages of 10000 random walks composed just by effective mutations ($MI > 0$). To support the analysis, the horizontal line in the chart displays an estimate of the phenotypic distance



(a) GE



(b) SGE

Fig. 14: Distribution of the phenotypic distances between an original solution and mutants iteratively generated over a random walk with 20 steps: (a): GE; (b): SGE. Results are averages of 10000 runs.

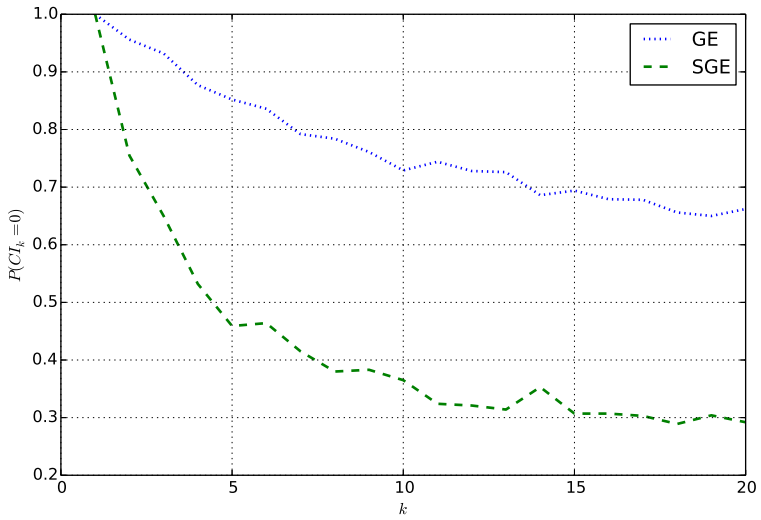


Fig. 15: Evolution of $P(CI = 0)$, as the genotypic distance between parents increases. Results are averages of 10000 runs.

between two random solutions (averaged over 10000 random pairs of individuals). Generically, as mutations accumulate, the resulting phenotypes gradually diverge from the derivation tree of the original solution. There are, however, important differences in what concerns the locality of the two representations. SGE promotes a gradual differentiation along the random walk, as every effective mutation slightly modifies the resulting phenotype. This behavior is illustrative of a representation with high locality and it supports an effective exploitation of neighbor solutions. When $k > 15$, the MI of SGE tends to stabilize. At this point, many individuals have reached the maximum possible size (given the predetermined recursion limit). Thus, most of the new trees are obtained by modifying leaf nodes labels and do not result from structural changes.

By contrast, effective mutations in GE immediately lead to the appearance of highly dissimilar derivation trees. The locality is clearly lower than that of SGE and, a small number of mutations obliterates the connection between solutions. Just after $k = 10$ mutation steps, the phenotypic distance is equivalent to the distance between two random solutions. The combined analysis of this result and the outcome of the previous section reveals that, while most mutations in GE are redundant, the few that are immediately effective compromise a meaningful exploration of the search space. The higher locality of SGE is a direct consequence of the genotype organization. By relying on a one-to-one correspondence between genes and non-terminals, one ensures that a mutation only modifies the derivation options of the mutated non-terminal, leaving the

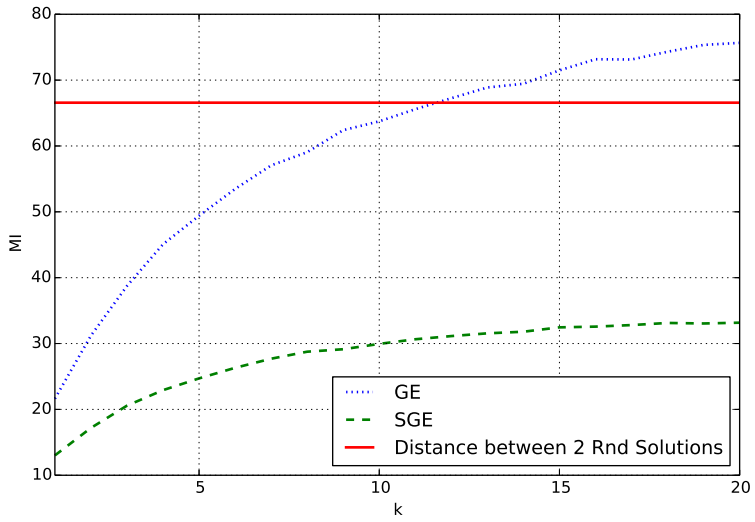


Fig. 16: Evolution of MI over a random walk with 20 steps, composed just by effective mutations. Results are averages of 10000 runs.

others unchanged. On the contrary, mutations in GE can modify the derivation options of several non-terminals, which may result in a substantially different tree.

To estimate how crossover impacts locality we repeated the experiment where we fixed one of the parents and obtained the other by applying successive mutations $k = 1, 2, 3, 4, ...20$, allowing the generation of increasing distant pairs. In this section we ignore the applications of crossover that results in $CI = 0$. Results presented are averages of 10000 repetitions. Fig. 17 presents the evolution of CI, as the genotypic distance between parents increases. As expected, for both representations there is a direct correlation between the dissimilarity of parents and the offspring innovation level. In accordance with the results obtained with mutation, SGE promotes a gradual and sustained CI growth, whereas standard GE is unable to avoid an abrupt rise since the first steps of the random walk.

6 Discussion

The analysis from last section helps to unveil why SGE has an increased performance, when compared to the standard GE representation. On the one hand, SGE is able to reduce the extremely high redundancy levels of GE, thus promoting search efficiency. Additionally, it exhibits a balanced locality, which fosters an appropriate sampling of the search space. There are several

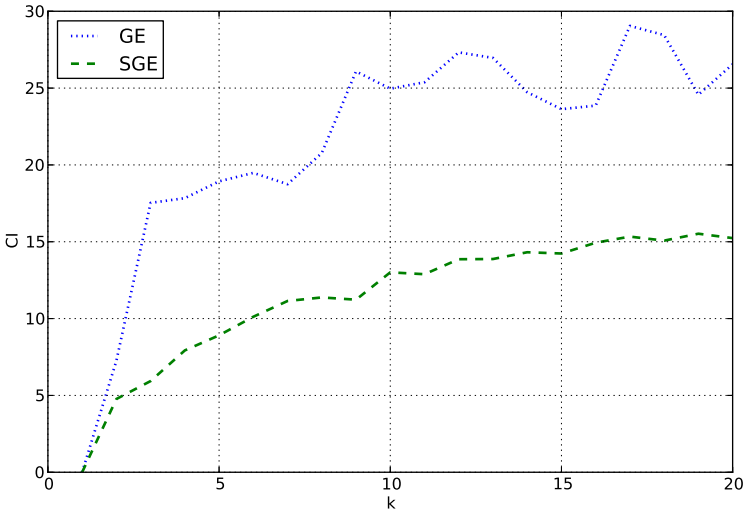


Fig. 17: Evolution of CI, as the genotypic distance between parents increases. The study considers only situations where $P(CI > 0)$ and results are averages of 10000 runs.

novel design options that differentiate SGE from the standard GE representation and, to complement the analysis, it is important to fully understand how these different components and settings impact SGE behavior.

The maximum recursion level is a key parameter to define when dealing with recursive grammars, as it determines the maximum tree size. If the limit is too low, then it might compromise the discovery of good quality solutions. This way the usual option is to define a safe value that allows for some redundancy, at the expense of having a larger search space to explore. To study the influence of this limit, we repeated the MI experiments detailed in section 5 with alternative levels of recursion: $\{4, 8, 10\}$. In what concerns redundancy, the percentage of mutations that do not lead to an immediate phenotypic change ranges between 33% for 4 recursion levels and approximately 50% for 10 recursion levels. As expected, redundancy slightly increases as more recursive productions are added to the grammar, but the values are still clearly bellow those achieved by standard GE. Fig. 18 summarizes the MI evolution for effective mutations, *i.e.*, mutations that immediately change the phenotype. A brief perusal of the results confirms that the recursion level is directly correlated to locality. The general trend is the same for all lines displayed in the chart, but there is a clear hierarchy in what concerns the level of recursion: higher values allow for a more pronounced departure from the starting solution. This is an expected outcome, as higher recursion levels enable the

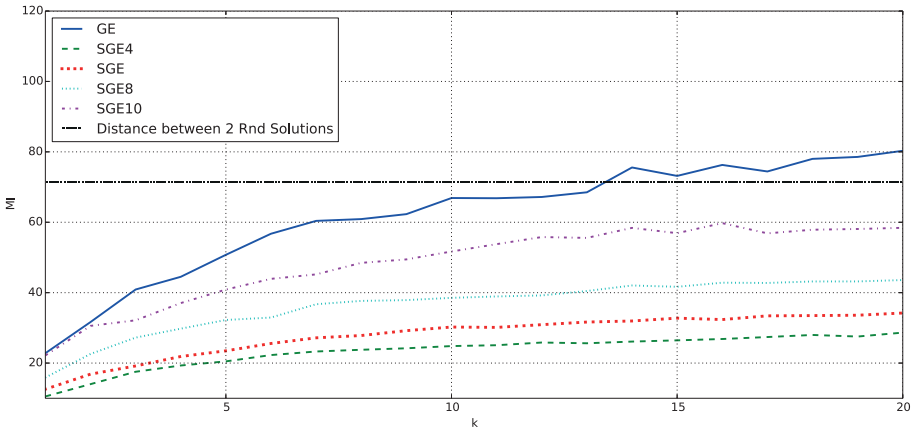


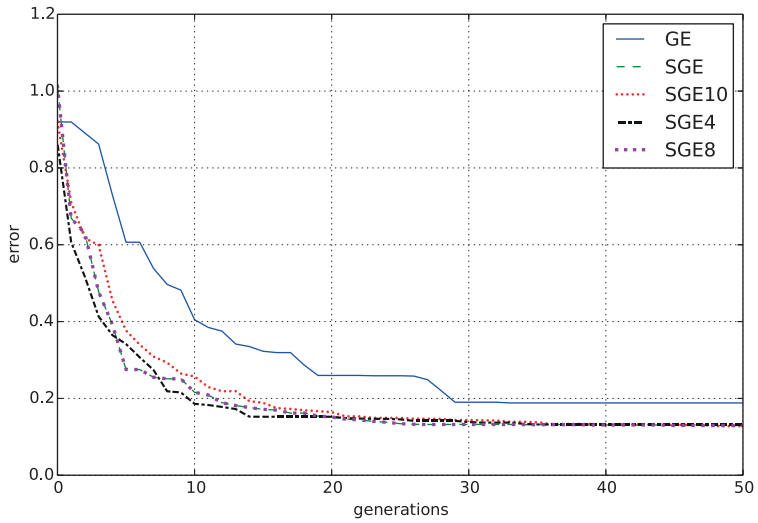
Fig. 18: Evolution of MI over a random walk with 20 steps, composed just by effective mutations. Different levels of recursion are considered for SGE: $\{4, 6, 8, 10\}$. Results are averages of 10000 runs.

appearance of larger derivation trees, thus creating room for higher degrees of innovation.

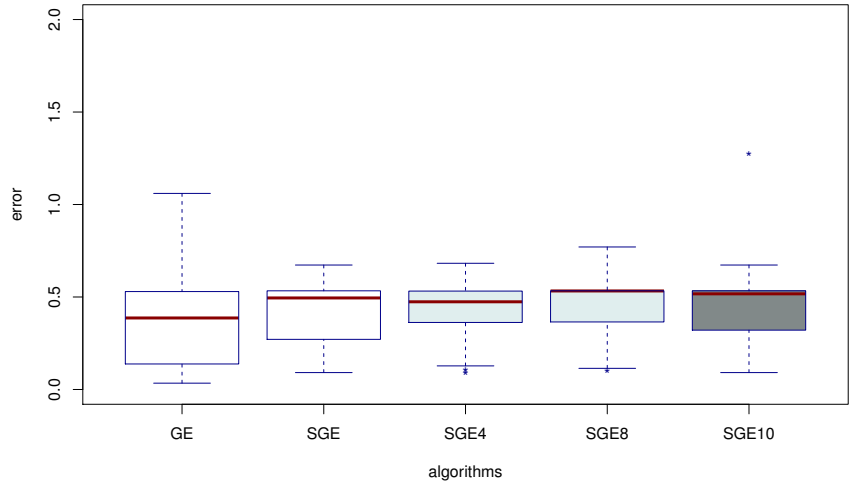
Given the MI disparity seen in Fig. 18, a question that immediately arises is how the variation in the recursion level, and as a direct consequence, in redundancy and locality, impacts SGE search effectiveness. To estimate this effect, we repeated all optimization experiments described in Section 4 with the additional recursion levels. Fig. 19 presents the results for the harmonic curve regression problem. The outcomes obtained by SGE with different recursion levels are similar and the small variations are never statistically significant. Results obtained in the other optimization problems selected for this study follow the same trend, suggesting that SGE is robust to moderate variations in the recursion level without compromising effectiveness.

It is worth mentioning that the specification of a hard limit for the number of recursive calls is an important prerequisite and should be considered as a shortcoming of the current SGE framework. Results presented in this section suggest that the definition of the maximum recursive level does not require a rigorous and precise fine-tuning. In any case, such specification must be done prior to the optimization, is common to all recursive productions, and then has an impact in the search performance. A valuable improvement to the current proposal would be the development of a flexible SGE representation with the ability to self-adapt the maximum recursion level of individual productions.

SGE new representation has a dual effect, as it minimizes redundancy while, at the same time, increases locality. We performed an additional test to analyze if any of these features is more important than the other in determining SGE behavior. Redundancy in standard GE is the result of non-expressed



(a) Optimization



(b) Generalization

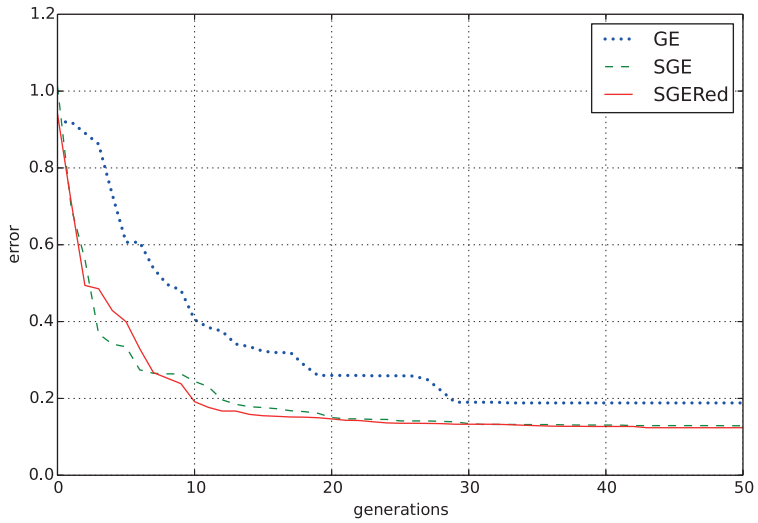
Fig. 19: Results obtained in the harmonic curve regression by SGE with different levels of recursion: (a) displays the evolution of the MBF; (b) presents the generalization results of the best models obtained in the last generation of optimization. Results are averages of 30 runs.

codons and of the application of the modulo rule to determine which option to use when expanding a non-terminal. SGE eliminates the last source of redundancy, while it keeps the first (it can also have non-expressed values in its genotype). We can easily create a redundant SGE variant (*SGERed*): when filling the lists belonging to the genes, integer values randomly drawn from [0,255] are selected and the number of options of the corresponding non-terminal are not considered. This way, SGERed also relies on the modulo rule to select the expansion alternative. Considering 6 recursion levels, SGERed redundancy after the application of one mutation raises to approximately 60%, above the 40% of SGE. The difference between the redundancy exhibited by SGERed and standard GE is now only a consequence of the genome size (for GE) and the recursion level (SGERed). The application of SGERed to an optimization task reveals an interesting feature. Fig. 20 displays the outcomes obtained in the harmonic curve regression problem. Results reveal that SGE and SGERed have an identical behavior, suggesting that locality is the main strength of SGE and is the key feature that allows the new representation to have an enhanced performance. The same trend is visible in other optimization problems.

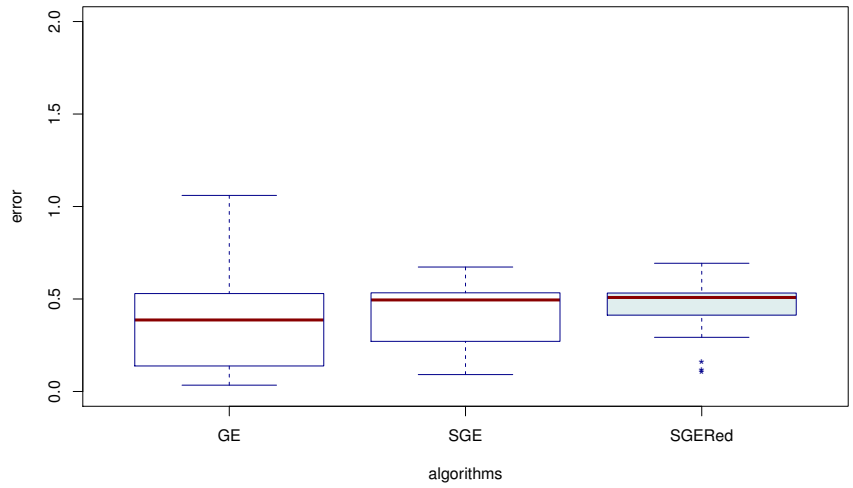
It might be argued that standard GE is unfairly penalized in the analysis, since it does not have an explicit way to perform a meaningful definition of the genome size. If the genotype has too many codons, then redundancy is extremely high. On the contrary, small genotypes heavily rely in wrapping, thus compromising locality. A final set of results reveals that, although this limitation exists, the main weakness of standard GE is its low locality.

We selected the grammar used by SGE for the harmonic curve regression problem after applying the pre-processing step (considering 6 levels of recursion). Given this grammar we can immediately define a fixed size for the GE genotype, as there are no recursive productions. Also, wrapping is not needed, since mapping never goes beyond the final codon. This GE variant, which we identify as GEFixed, differs from SGE in two issues: the selection of an option when expanding a non-terminal and the non-existence of a direct correspondence between genes and non-terminals. We recall that these are the two key novel issues addressed in the definition of SGE.

Fig. 21 presents the results obtained by SGE and GEFixed in the harmonic curve regression. The behavior of GEFixed is slightly better than that of standard GE. However, it is still outperformed by SGE in the optimization phase (differences are statistically significant). On the generalization step, the behaviors are equivalent (just like in Section 4). These final results reinforce that the key factor for SGE success is its structured genotypic definition that allows for an increased locality of the representation and for a sustained reduction of redundancy.

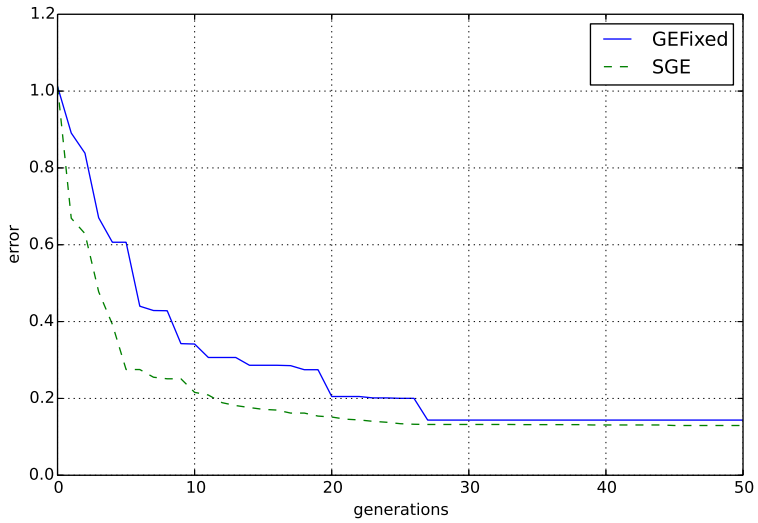


(a) Optimization

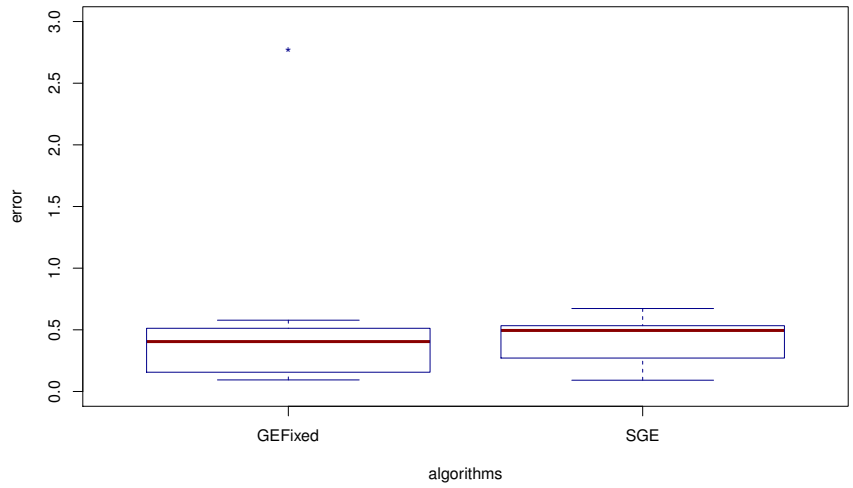


(b) Generalization

Fig. 20: Results obtained in the harmonic curve regression, considering also *SGERed*, a redundant version of SGE: (a) displays the evolution of the MBF; (b) presents the generalization results of the best models obtained in the last generation of optimization. Results are averages of 30 runs.



(a) Optimization



(b) Generalization

Fig. 21: Results obtained in the harmonic curve regression, considering *GEFixed*, a GE variant with a fixed-sized genome: (a) displays the evolution of the MBF; (b) presents the generalization results of the best models obtained in the last generation of optimization. Results are averages of 30 runs.

7 Conclusions

SGE is a new genotypic representation for GE that explicitly considers the features of the grammar being used. The definition of the genotype requires the application of a pre-processing step where recursive productions are rewritten in a non-recursive format. After pre-processing is over, the structured genotype is defined. Each gene links to a specific non-terminal and it encodes a list of integers that help to determine the derivation options during mapping.

Standard GE approaches have been applied to different optimization scenarios over the past 15 years. Despite their success, recent works have identified weaknesses in the GE representation that compromise a meaningful exploration of the search space [32,37]. SGE is a relevant step towards the development of an improved structured representation for GE. Its effectiveness was tested on an extended set of benchmarks. Results were encouraging, as SGE outperformed standard GE in the optimization/training stage of all selected problems. Additionally, the optimization effectiveness does not compromise the generalization ability of evolved solutions, as confirmed by results presented in Section 4.

SGE was defined aiming at increasing the locality and also at reducing the extremely high redundancy levels that are usually observed in standard GE representations. We considered the framework proposed by Raidl et al. [29] that relies on a set of static measures to determine the innovation of variation operators, and thus, estimate the redundancy and locality of a given representation. Empirical results confirm that SGE clearly reduces the redundancy level of standard GE, thus fostering search efficiency. When dealing with recursive grammars, SGE redundancy is directly correlated with the recursion level. However, since SGE does not rely on the modulo rule to perform mapping, redundancy is always clearly lower than the extremely high values exhibited by standard GE.

The locality of SGE was studied by considering the impact of effective genotypic variations in the phenotype. Results show that the new representation exhibits a balanced locality, as the genotypic changes that are iteratively accumulated, promote a gradual modification in the phenotypes. This behavior is in contrast with standard GE, where small changes in the genotype tend to create abrupt modifications in the derivation tree. SGE high locality fosters a meaningful exploration of the search space, providing an explanation for the increased optimization effectiveness. In Section 6 we defined several GE and SGE variants and provided a set of complementary results revealing that, although redundancy is useful to enhance search efficiency, the key feature that justifies the SGE optimization behavior is its ability to increase locality.

There are plenty of possible extensions for the current work. On the one hand, we believe that the original SGE representation proposed in this paper might be further enhanced, *e.g.*, by considering an adaptive mechanism that automatically adjusts the maximum number of recursive calls. This will simplify the task of non-expert practitioners when applying SGE to optimization situations. Also, SGE optimization behavior should be compared with other

GP variants. Results presented in Section 4 confirm that SGE improves over the standard GE representation, but it is essential to expand the comparative study to consider other grammar and non-grammar based GP extensions. Finally, the analysis presented here is based on static measures of innovation. In the near future we aim to extend it to a dynamic scenario, to study how the interplay between different algorithmic components (variation operators, settings, selective pressure) impacts the evolution of locality and redundancy throughout an optimization run.

Acknowledgements The first author is funded by Fundação para a Ciência e Tecnologia (FCT), Portugal, under the grant SFRH/BD/79649/2011.

References

1. Archetti, F., Lanzeni, S., Messina, E., Vanneschi, L.: Genetic programming for computational pharmacokinetics in drug discovery and development. Genetic Programming and Evolvable Machines **8**(4), 413–432 (2007)
2. Azad, R.M.A.: A position independent representation for evolutionary automatic programming algorithms - the chorus system. Ph.D. thesis, University of Limerick, Ireland (2003)
3. Brameier, M., Banzhaf, W.: Explicit control of diversity and effective variation distance in linear genetic programming. In: Genetic Programming, *Lecture Notes in Computer Science*, vol. 2278, pp. 37–49. Springer (2002)
4. Byrne, J., O'Neill, M., Brabazon, A.: Structural and nodal mutation in grammatical evolution. In: Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation, pp. 1881–1882. New York, NY, USA (2009)
5. Byrne, J., O'Neill, M., McDermott, J., Brabazon, A.: An analysis of the behaviour of mutation in grammatical evolution. In: Genetic Programming, *Lecture Notes in Computer Science*, vol. 6021, pp. 14–25. Springer (2010)
6. Castle, T., Johnson, C.: Positional effect of crossover and mutation in grammatical evolution. In: Genetic Programming, *Lecture Notes in Computer Science*, vol. 6021, pp. 26–37. Springer (2010)
7. Dempsey, I., O'Neill, M., Brabazon, A.: Foundations in grammatical evolution for dynamic environments. Springer (2009)
8. Fagan, D.: An analysis of genotype-phenotype mapping in grammatical evolution (2014)
9. Fagan, D., O'Neill, M., Galván-López, E., Brabazon, A., McGarraghy, S.: An analysis of genotype-phenotype maps in grammatical evolution. In: Genetic Programming, *Lecture Notes in Computer Science*, vol. 6021, pp. 62–73. Springer (2010)
10. Field, A.: Discovering statistics using IBM SPSS statistics. Sage (2013)
11. Gonçalves, I., Silva, S.: Balancing learning and overfitting in genetic programming with interleaved sampling of training data. In: Genetic Programming, *Lecture Notes in Computer Science*, vol. 7831, pp. 73–84. Springer (2013)
12. Gottlieb, J., Eckert, C.: A comparison of two representations for the fixed charge transportation problem. In: Parallel Problem Solving from Nature PPSN VI, *Lecture Notes in Computer Science*, vol. 1917, pp. 345–354. Springer (2000)
13. Gottlieb, J., Raidl, G.: Characterizing locality in decoder-based eas for the multidimensional knapsack problem. In: Artificial Evolution, *Lecture Notes in Computer Science*, vol. 1829, pp. 38–52. Springer (2000)
14. Harper, R.: Spatial co-evolution: quicker, fitter and less bloated. In: Proceedings of the 14th annual conference on Genetic and evolutionary computation, pp. 759–766. ACM (2012)
15. Hugosson, J., Hemberg, E., Brabazon, A., O'Neill, M.: Genotype representations in grammatical evolution. Applied Soft Computing **10**(1), 36 – 43 (2010)

16. Jones, T., Forrest, S.: Fitness distance correlation as a measure of problem difficulty for genetic algorithms. In: Proceedings of the 6th International Conference on Genetic Algorithms, pp. 184–192. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1995)
17. Keijzer, M., O'Neill, M., Ryan, C., Cattolico, M.: Grammatical evolution rules: The mod and the bucket rule. In: Genetic Programming, *Lecture Notes in Computer Science*, vol. 2278, pp. 123–130. Springer (2002)
18. Keller, R., Banzhaf, W.: Genetic programming using genotype-phenotype mapping from linear genomes into linear phenotypes. In: Proceedings of the 1st Annual Conference on Genetic Programming, pp. 116–122. MIT Press, Cambridge, MA, USA (1996)
19. Koza, J.R.: Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press, Cambridge, MA, USA (1992)
20. Lourenço, N., Pereira, F.B., Costa, E.: SGE : A structured representation for grammatical evolution. In: 12th International Conference Evolution Artificielle, EA 2015. Lyon, France (2015)
21. McKay, R.I., Hoai, N.X., Whigham, P., Shan, Y., O'Neill, M.: Grammar-based Genetic Programming: a survey. Genetic Programming and Evolvable Machines **11**(3-4) (2010)
22. O'Neill, M., Brabazon, A.: Grammatical differential evolution. In: H.R. Arabnia (ed.) Proceedings of the 2006 International Conference on Artificial Intelligence, ICAI 2006, vol. 1, pp. 231–236 (2006)
23. O'Neill, M., Brabazon, A.: Grammatical swarm: The generation of programs by social programming. Natural Computing **5**(4), 443–462 (2006)
24. O'Neill, M., Brabazon, A., Nicolau, M., McGarraghy, S., Keenan, P.: pigrammatical evolution. In: Genetic and Evolutionary Computation, vol. 3103, pp. 617–629. Springer (2004)
25. O'Neill, M., Hemberg, E., Gilligan, C., Bartley, E., McDermott, J., Brabazon, A.: Geva - grammatical evolution in java (v 2.0). Tech. rep., Technical Report, UCD School of Computer Science (2008)
26. O'Neill, M., Ryan, C.: Grammatical Evolution: Evolutionary Automatic Programming in an Arbitrary Language. Kluwer Academic Publishers, Norwell, MA, USA (2003)
27. O'Reilly, U.M.: Using a distance metric on genetic programs to understand genetic operators. In: IEEE International Conference on Systems, Man, and Cybernetics, vol. 5, pp. 4092–4097 vol.5 (1997)
28. O'Sullivan, J., Ryan, C.: An investigation into the use of different search strategies with grammatical evolution. In: Genetic Programming, pp. 268–277. Springer (2002)
29. Raidl, G., Gottlieb, J.: Empirical analysis of locality, heritability and heuristic bias in evolutionary algorithms: A case study for the multidimensional knapsack problem. Evolutionary Computation **13**(4), 441–475 (2005)
30. Rothlauf, F.: On the locality of representations. In: Genetic and Evolutionary Computation, *Lecture Notes in Computer Science*, vol. 2724, pp. 1608–1609. Springer (2003)
31. Rothlauf, F.: Representations for genetic and evolutionary algorithms. Springer (2006)
32. Rothlauf, F., Oetzel, M.: On the locality of grammatical evolution. In: Genetic Programming, Lecture Notes in Computer Science, pp. 320–330. Springer (2006)
33. Ryan, C., Azad, A.: Sensible initialisation in grammatical evolution. In: A.M. Barry (ed.) GECCO 2003: Proceedings of the Bird of a Feather Workshops, Genetic and Evolutionary Computation Conference, pp. 142–145. AAAI (2003)
34. Ryan, C., Azad, A., Sheahan, A., O'Neill, M.: No coercion and no prohibition, a position independent encoding scheme for evolutionary algorithms–the chorus system. In: Genetic Programming, pp. 131–141. Springer (2002)
35. Ryan, C., Collins, J., O'Neill, M.: Grammatical evolution: Evolving programs for an arbitrary language. In: W. Banzhaf, R. Poli, M. Schoenauer, T. Fogarty (eds.) Genetic Programming, *Lecture Notes in Computer Science*, vol. 1391, pp. 83–96. Springer (1998)
36. Thorhauer, A., Rothlauf, F.: On the locality of standard search operators in grammatical evolution. In: Parallel Problem Solving from Nature - PPSN XIII, *Lecture Notes in Computer Science*, vol. 8672, pp. 465–475. Springer International Publishing (2014)
37. Whigham, P.A., Dick, G., Maclaurin, J., Owen, C.A.: Examining the best of both worlds of grammatical evolution. In: Proceedings of the 2015 on Genetic and Evolutionary Computation Conference, pp. 1111–1118. ACM (2015)

38. White, D., McDermott, J., Castelli, M., Manzoni, L., Goldman, B., Kronberger, G., Jaskowski, W., O'Reilly, U.M., Luke, S.: Better gp benchmarks: community survey results and proposals. Genetic Programming and Evolvable Machines **14**(1), 3–29 (2013)
39. Zhang, K., Shasha, D.: Simple fast algorithms for the editing distance between trees and related problems. SIAM Journal on Computing **18**(6), 1245–1262 (1989)