Evolvability in Grammatical Evolution

Eric Medvet
DIA - Università di Trieste
Trieste, Italy
emedvet@units.it

Fabio Daolio
University of Stirling
Stirling, Scotland
fda@cs.stir.ac.uk

Danny Tagliapietra
DIA - Università di Trieste
Trieste, Italy
danny.tagliapietra@gmail.com

ABSTRACT

Evolvability is a measure of the ability of an Evolutionary Algorithm (EA) to improve the fitness of an individual when applying a genetic operator. Other than the specific problem, many aspects of the EA may impact on the evolvability, most notably the genetic operators and, if present, the genotype-phenotype mapping function. Grammatical Evolution (GE) is an EA in which the mapping function plays a crucial role since it allows to map any binary genotype into a program expressed in any user-provided language, defined by a context-free grammar. While GE mapping favored a successful application of GE to many different problems, it has also been criticized for scarcely adhering to the variational inheritance principle, which itself may hamper GE evolvability. In this paper, we experimentally study GE evolvability in different conditions, that is, problems, mapping functions, genotype sizes, and genetic operators. Results suggest that there is not a single factor determining GE evolvability: in particular, the mapping function alone does not deliver better evolvability regardless of the problem. Instead, GE redundancy, which itself is the result of the combined effect of several factors, has a strong impact on the evolvability.

CCS CONCEPTS

•**Theory of computation → Genetic programming;** *Grammars and context-free languages;* •**Computing methodologies → Heuristic function construction;**

KEYWORDS

Locality, Fitness-landscape, Genotype-phenotype mapping

ACM Reference format:
Eric Medvet, Fabio Daolio, and Danny Tagliapietra. 2017. Evolvability in Grammatical Evolution. In *Proceedings of The Genetic and Evolutionary Computation Conference, Berlin, Germany, July 15–19, 2017 (GECCO'17),* 8 pages.
DOI: http://dx.doi.org/10.1145/3071178.3071298

1 INTRODUCTION

Grammatical Evolution (GE) can be regarded as a form of Genetic Programming (GP) that uses an indirect representation. As in GP, the evolutionary algorithm evolves programs to solve a given problem or task. However, GE programs are not directly encoded as trees; instead, individuals are represented through binary strings, the *genotype*, that are successively translated into programs, the *phenotype*, following the rules of a problem-specific grammar. The programs execution then allows one to assign a score, the *fitness*, to each candidate solution. With such a complex genotype-phenotype-fitness mapping comes a great flexibility, which requires the researcher to take several design decisions. E.g., how to choose the most appropriate genotype size? [2]

This has been and still is a common situation in the field of Evolutionary Computation. As Culberson [4] put it: "the researcher trying to solve a problem is then placed in the unfortunate position of having to find a representation, operators and parameter settings to make a poorly understood system solve a poorly understood problem". Fitness landscape analysis offers a possible way around this, by providing empirical measures to characterise problem hardness from the point of view of search heuristics. Indeed, fitness landscape analysis aims to improve problem understanding and to inform the choice and the design of Metaheuristics [13]. This paper precisely focuses on landscape measures that are related to the concept of *evolvability*, as the ability of a population-based Metaheuristic to produce offsprings that are fitter than their parents. In particular, we investigate the evolvability of different genotype-phenotype mappers in the context of Grammatical Evolution. We then seek to interpret our findings in the light of other metrics, namely *redundancy* and *locality*, that can be used to characterise a mapper and hence a GE variant.

The remaining of the paper is organized as follows. Section 2 briefly overviews the context and the state-of-the-art. Section 3 presents the evolvability framework adopted in the present study. Section 4 describes the different GE variants included in this analysis. Section 5 reports the experimental results and our interpretation. Finally, Section 6 draws the concluding remarks.

2 RELATED WORK

A number of empirical studies have been aiming to characterise GE behaviour and the interplay between representation and variation operators, especially in terms of locality and redundancy [3, 11, 18]. For instance, among more recent work, Thorhauer has investigated representation bias in GE for binary trees, and found that redundancy is non-uniform, with large discrepancies in the number of genotypes that encode the same phenotype [21]. Medvet et al. have performed a comparative analysis of locality and redundancy along the evolutionary process of several GE variants, and found that genotype size highly impacts redundancy in the dynamic scenario [14]. However, the bigger picture is still unclear.

Literature on Genetic Programming and its fitness landscape could offer another perspective. In any evolutionary algorithm, the ability of a population to generate fitter individuals is paramount;

GECCO'17, Berlin, Germany

DOI: http://dx.doi.org/10.1145/3071178.3071298

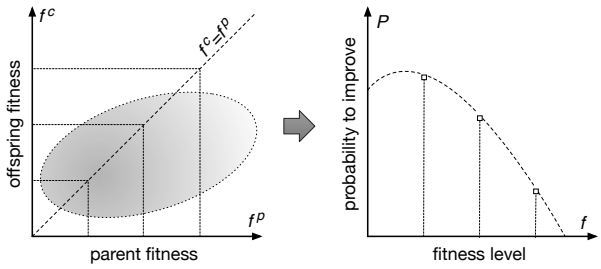

Figure 1: From Fitness Cloud to Fitness-Probability Cloud.

this concept of evolvability can provide a relevant and general-enough framework that hopefully brings new insight into problem hardness for GE. Altenberg has shown in the 1990s how, in the GP evolution of tree-based programs, representation and variation operators must interact to obtain evolvability, and how evolvability itself changes over time [1]. In the present work, since we aim to focus on the extra representation layer that GE adds, that is, the mapper, aside from other algorithmic components, we neglect the dynamical aspects and only study the effect of variation operators on randomly-generated individuals. The simplifying assumption we make is that the variational aspect should largely determine the evolutionary dynamics, as in the case of traditional GP [1].

Among the evolvability metrics [13], we refer to the Fitness Cloud framework introduced by Verel et al [24], which essentially depicts evolvability in terms of the correlation between the fitness value of a parent and that of its offsprings. Following this, we adopt the recent Fitness-Probability Cloud proposed by Lu et al. [12], which in essence gauges the correlation between the fitness of a parent and the rate with which its offspring could improve upon it. Figure 1 provides a visual abstract; the necessary definitions are given in the next section.

3 EVOLVABILITY

In combinatorial optimisation, a *Fitness Landscape* can be formalised as a triple (X, N, f), where X is the set of candidate solutions, $N : X \mapsto 2^X$ is the neighbourhood induced by a given search operator op, $N(x) = \{x' \mid x' = \mathrm{op}(x)\}$, and $f : X \mapsto \mathbb{R}$ is the fitness function that maps each solution x to its fitness value $f(x)$.

Given a sample Γ of candidate parent solutions γ_i with fitness values $f_i = f(\gamma_i)$, for each $\gamma_i \in \Gamma$ we can generate K neighbours or offsprings by repeatedly applying the genetic operator op. The *Fitness Cloud* (FC) is then defined on the set of pairs $\{(f^p, f^c) \mid f^p = f(\gamma_i) \wedge f^c = f(\mathrm{op}(\gamma_i))\}$. Note that, in the case of the crossover operator, a pair of parents is needed but only the best one is considered for the FC composition: $f^p = \max(f^{p_1}, f^{p_2})$.

If we partition the observed fitness values into m contiguous bins or *levels* by considering increasing fitness thresholds $\{f_0, f_1, \ldots, f_m\}$, then from the fitness cloud we can estimate the *Escape Probability*. That is, for each level i, the expected probability to improve upon f_i after applying op once. It can be empirically estimated by the relative frequency of offsprings that improve upon their parents in level i. We denote this quantity by P_i. The *Fitness-Probability Cloud* (FPC) is then the set of pairs $\{(f_i, P_i)\}$.

From the FPC, a single evolvability metric can be derived: the *Accumulated Escape Probability* (AEP), which in this work is simply the average escape probability over all considered levels. Figure 1 provides a schematic view of these concepts.

4 GE VARIANTS

Since the genotype-phenotype mapping function is the peculiar component of GE, we included in our analysis different alternatives for this component, corresponding to different variants of the original GE: breadth-first GE (BGE), πGE, and SGE—we did not include few other variants we were aware of (such as, e.g. [9, 19]), because they introduced minor changes over standard GE or did not result in relevant improvements.

Three of the four variants (GE, BGE, and πGE) differ only in the mapping function which, in all cases, operates on binary genotypes, that is, variable-length bit strings. The mapping function of SGE, instead, operates on genotypes consisting of fixed-length integer strings and, as a consequence, employs specific genetic operators. In this section, we describe the 4 considered variants with a particular focus on their genotype-phenotype mapping procedure. We will denote by $\mathcal{G} = (N, T, s_0, R)$ the Context-Free Grammar (CFG), where N denotes the non-empty set of non-terminal symbols, T denotes the non-empty set of terminal symbols, $s_0 \in N$ denotes the starting symbol (or axiom), and R denotes the set of production rules; by p the phenotype, that is, a string of the language $\mathcal{L}(\mathcal{G})$ defined by $\mathcal{G}$; and by g the genotype.

4.1 Standard GE

In its original form, proposed by Ryan, Collins, and O'Neill in 1998 [20], the genotype g is seen as a sequence of integers, each one termed *codon* and consisting of n consecutive bits. The parameter n is conventionally set to 8; however, in some applications (e.g., [2]), it has been set to a value that is tailored to the specific grammar.

The genotype-phenotype mapping function of GE is an iterative procedure which starts with the phenotype $p = s_0$, i.e., to the grammar starting symbol, a counter $i = 0$, and a counter $w = 0$. Then, the following steps are iterated. (1) The leftmost non-terminal s in p is expanded by using the j-th option (zero-based indexing) in the production rule r_s for s in $\mathcal{G}$. The value of j is set to the remainder of the division between the value g_i of the i-th codon (zero-based indexing) and the number $|r_s|$ of options in r_s, i.e., $j = g_i \bmod |r_s|$. (2) The counter i is incremented; if i exceeds the number of codons, i.e., if $i > \frac{|g|}{n}$, then i is set to 0 and w is incremented—the latter operation is called *wrapping* and w represents the number of wraps performed during the mapping. (3) If w exceeds a predefined threshold n_w, then the procedure is stopped and p is set to a *null phenotype* whose fitness will be set, conventionally, to the worst possible fitness value. The procedure is iterated until no more non-terminals exist in p.

The rationale for the wrapping, which in practice corresponds to reuse the genotype, when needed, is to extend the applicability of GE to recursive grammars, that is, to languages containing non-finite strings. However, an upper bound n_w to the number of wrapping operations must be enforced to avoid an endless mapping. The grammar complexity, the upper bound n_w, and the genotype

```
<expr> ::= ( <expr> <op> <expr> ) |
           ( <pre-op> <expr> ) | <var>
  <op> ::= + | *
<pre-op> ::= uminus | 1/ | sqrt
 <var> ::= x
```

(a) Harmonic

```
<expr> ::= ( <expr> <op> <expr> ) |
           ( <pre-op> <expr> ) | <var>
  <op> ::= + | - | * | /
<pre-op> ::= sin | cos | exp | log
 <var> ::= x | 1.0
```

(b) Polynomial

```
<code> ::= <line> | <code> <line>
<line> ::= <if> | <op>
  <if> ::= if(food_ahead()) <line> else <line>
  <op> ::= left(); | right(); | move();
```

(c) Santa-Fe

```
    <text> ::= <sentence> ␣ <text> | <sentence>
<sentence> ::= <Word> ␣ <sentence> |
               <word> ␣ <sentence> |
               <word> <punct>
    <word> ::= <letter> <word> | <letter>
    <Word> ::= <Letter> <word>
  <letter> ::= <vowel> | <consonant>
   <vowel> ::= a | o | u | e | i
<consonant> ::= b | c | d | f | g | h | j | k | l |
               m | n | p | q | r | s | t | v | w |
               x | y | z
  <Letter> ::= <Vowel> | <Consonant>
   <Vowel> ::= A | O | U | E | I
<Consonant> ::= B | C | D | F | G | H | J | K | L |
               M | N | P | Q | R | S | T | V | W |
               X | Y | Z
   <punct> ::= ! | ? | .
```

(d) Text

Figure 2: The grammars of the considered problems.

size $|g|$ are clearly related and choosing an approrpiate value for the latter is not straightforward.

Many different approaches have been proposed concerning the EA components other than the mapping function in GE [15, 17]. In this work, we do not run an evolution; the only componenent which is relevant to our work is hence the set of genetic operators. We used the bit flip mutation (in which each bit in the phenotype may be flipped according to a predefined probability p_{mut}) and the one-point crossover (in which the cut points on the two parents are chosen independently and hence the length of the resulting child can be different from parents length).

4.2 Breadth-first GE (BGE)

In [6], Fagan et al. compared different mapping functions and introduced a new variant called Breadth-first GE (BGE). The only different between BGE and GE is in step 1 of the iterative procedure, in which the least deep (closest to the tree root) non-terminal in p is chosen to be expanded, instead of the leftmost non-terminal. As shown experimentally in [6], BGE is not significantly better (or worse) than GE, but we consider it in our study because of the rather different way in which phenotypes are grown according to this mapping function.

4.3 Position-independent GE (πGE)

In both the standard GE mapping and BGE mapping, the non-terminal to be expanded is chosen with a predefined criterion. According to [16], this design does not foster the arising of building blocks in the genotype, that is, small sequences of codons which, upon mapping, correspond to useful sequences of symbols in the phenotype. In order to address this limitation, O. Neill et al. proposed in [16] the Position-independent GE (πGE), in which the choice of the non-terminal to be expanded and the choice of the specific expansion are decoupled.

Precisely, in πGE each codon consists of a pair $g_i^{\text{nont}}, g_i^{\text{rule}}$ integers, each of n bits—conventionally, n is set to 8. The mapping procedure differs from that of GE in step 1: in πGE, the non-terminal of p to be expanded is the j^{nont}-th one, with $j^{\text{nont}} = g_i^{\text{nont}} \mod n_s$, n_s being the number of non-terminals in p. Then, the rule option to be used is determined, as in standard GE, with $j^{\text{rule}} = g_i^{\text{rule}} \mod |r_s|$. As a consequence of this difference, in πGE the positions of the phenotype that are to be expanded are encoded in the genotype and evolve independently from corresponding expansion options.

4.4 Structural GE (SGE)

Structural GE (SGE) [10] is the most recent variant of GE. Differently than in standard GE, BGE, and πGE, in SGE the genotype is not a flat sequence of bits but it has a structure such that, during the mapping, each codon is used at most once and for choosing the expansion of a predefined non-terminal. The aim for this design choice is, according to the authors, to increase locality and decrease redundancy [11]. Since there is not a mechanism for reusing the genotype, SGE mapping does not apply to recursive grammars. However, the authors of [10] briefly describe a procedure for transforming any possibly recursive grammar G into a non-recursive grammar G' by imposing a maximum tree depth $d_{\max}$, a parameter for which is clearly not easy to find in advance an optimal value.

In detail, the genotype g in SGE is a fixed-length integer string which is composed of $|N|$ substrings, that is, one substring g_s for each non-terminal $s \in N$ of the grammar G. The length of each substring g_s is determined by the maximum number of expansions which can be applied to the corresponding non-terminal s according to the non-recursive grammar G'; the domain of each codon is set to $\{0, \ldots, |r_s| - 1\}$, r_s being the production rule for s.

The mapping function of SGE is an iterative procedure in which, initially, the phenotype is set to $p = s_0$, and a counter i_s for each non-terminal $s \in N$ is set to 0. The following steps are then iterated. (1) The leftmost non-terminal s in p is expanded by using the g_{s,i_s}-th option (zero-based indexing) of the rule r_s, with g_{s,i_s} denoting

the value of the i_s-th codon (zero-based indexing) in g_s. (2) The counter i_s is incremented. The procedure is iterated until no more non-terminals exist in p. It can be noted than SGE never maps a genotype to a null phenotype.

SGE uses genetic operators which are tailored to the structure of the genotype. In particular, the mutation consists in changing, with a probability p_{mut}, the value of each codon to a new random value in the appropriate domain. The crossover consists in exchanging the substrings g_s^1, g_s^2 of the parent genotypes corresponding to each non-terminal s in a randomly chosen subset $N' \subseteq N$.

5 EXPERIMENTAL ANALYSIS AND DISCUSSION

5.1 Benchmark problems and procedure

In order to perform a meaningful analysis of the evolvability of GE, we considered 4 problems: Harmonic, Polynomial, Santa-Fe, and Text. Three of them are classic benchmark problems for Genetic Programming and Grammatical Evolution [7, 25], whereas the latter (Text) has been introduced in [14] purposely for studying the properties of locality and redundancy in GE. The problems are briefly described below and their grammars are shown in Figure 2. We think that these 4 benchmark well represent real world problems, due to their diverse grammar complexities and fitness functions.

Harmonic In this symbolic regression problem, the goal is to approximate the function $f(x) = \sum_i^x \frac{1}{i}$ and the fitness is the sum of absolute errors computed in the points $x \in \{1, \dots, 50\}$.

Polynomial As for the Harmonic problem, the goal is to approximate the function $f(x) = x^4 + x^3 + x^2 + x$ and the fitness is computed in the points $x \in \{-1, -0.9, \dots, 0.9, 1\}$.

Santa-Fe The goal is to find a program which guides an artificial ant to collect 89 statically placed food items in a 32×32 grid within a maximum number of steps. The fitness is the number of missed food items.

Text The goal is to build a statically defined target string (`Hello world!` in the present paper) and the fitness is the edit distance between the target string and the string encoded by the individual.

We performed our analysis according to a procedure that closely resembles the one described in [12]; namely:

(1) For each GE variant, each problem, each genetic operator, and each genotype size $|g| \in \{128, 256, 512, 1024\}$ (with the exception of SGE, in which the genotype size is determined by the grammar), we randomly generated 300 genotypes g^p (for the mutation) or pairs (g^{p_1}, g^{p_2}) of genotypes (for the crossover).
(2) Then, for each genotype or pair of genotypes, we applied 30 times the genetic operator, obtaining each time a child genotype g^c.
(3) We then mapped the parent and child genotypes g^{p_1}, g^{p_2}, g^c to the corresponding phenotypes p^{p_1}, p^{p_2}, p^c and computed the corresponding fitness values f^{p_1}, f^{p_2}, f^c.

We possibly repeated the steps to ensure that, for GE, BGE, and πGE, all the three mapping resulted in a non-null phenotype. Besides avoiding null phenotypes, we did not take special arrangements while generating the genotypes for, e.g., favoring individuals with better fitness values.

Concerning the parameters of the mapping functions and of the genetic operators, we set: $n = 8$ and $n_w = 5$ for GE, BGE, and πGE; $d_{\max} = 6$ for SGE (as suggested by the authors in [10]); and $p_{\text{mut}} = 0.01$ for all variants.

Table 1: AEP for the mutation operator.

Problem	$\lvert g\rvert$	GE	BGE	πGE	SGE
Harmonic	75				0.018
	128	0.067	0.073	0.067	
	256	0.08	0.092	0.106	
	512	0.114	0.102	0.111	
	1024	0.113	0.11	0.12	
Polynomial	121				0.019
	128	0.047	0.048	0.049	
	256	0.067	0.06	0.074	
	512	0.071	0.072	0.071	
	1024	0.065	0.065	0.076	
Santa-Fe	31				0.008
	128	0.066	0.072	0.054	
	256	0.071	0.074	0.084	
	512	0.091	0.112	0.114	
	1024	0.123	0.141	0.148	
Text	85				0.011
	128	0.118	0.146	0.137	
	256	0.165	0.218	0.272	
	512	0.173	0.224	0.314	
	1024	0.203	0.223	0.334	

5.2 Accumulated Escape Probability

Tables 1 and 2 present the results, in terms of the AEP computed separately for each combination of GE variant, problem, and genotype size, respectively for the mutation and the crossover—for SGE, we express $|g|$ in bits by assuming for each codon the lowest number of bits required to encode the corresponding domain. The same results are also plotted in Figure 3, which shows AEP vs. the genotypes size $|g|$. Each number in the two tables and point in the figure is obtained by computing AEP on the 300×30 tuples (f^{p_1}, f^{p_2}, f^c) of fitness values.

Three interesting observations may be done. First, it can be seen that SGE greatly differs from the other three variants (GE, BGE, and πGE). In particular, SGE exhibits larger values of AEP with the crossover (with the exceptions of the Santa-Fe and Text problems) and smaller values with the mutation. This finding may be partly explained by the fact that the differences among GE, BGE, and πGE are negligible w.r.t. the differences between SGE and each of those three variants. Interestingly, it can also be observed that in the Text problem the differences among GE, BGE, and πGE are magnified.

Second, the impact of the genotype size is, at least for GE, BGE, and πGE, opposite for mutation and crossover. For the former, the larger the genotype, the greater the AEP; for the latter, the opposite. Since in SGE the genotype size $|g|$ is determined by the grammar

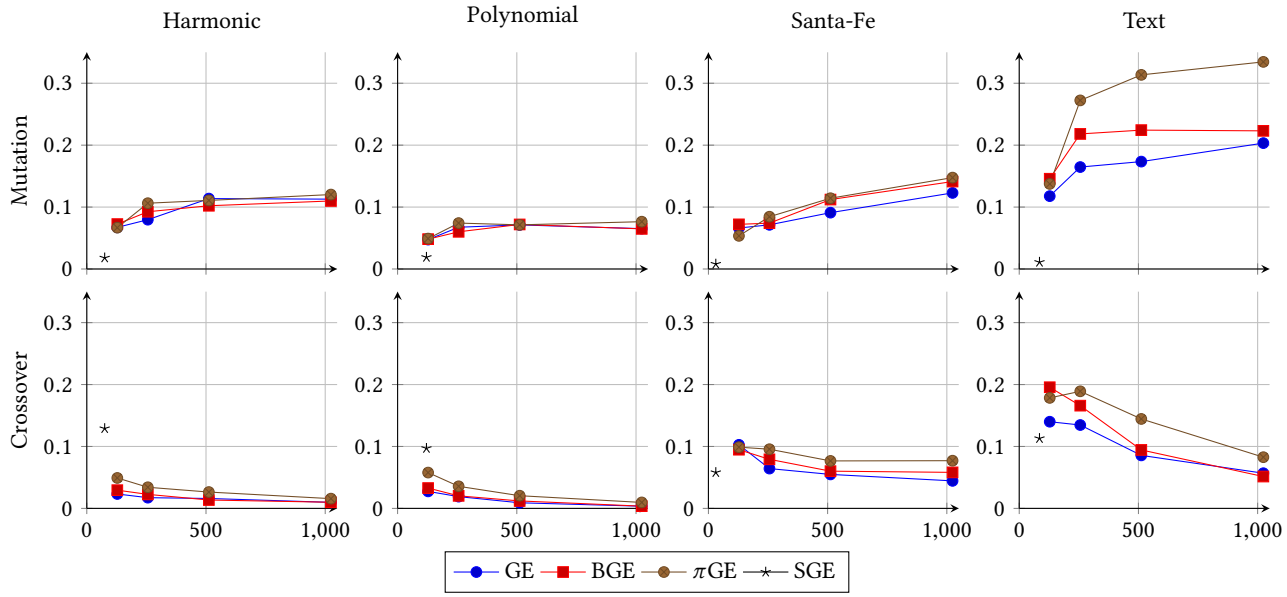

Figure 3: AEP vs. genotype size $|g|$, with different genetic operators (row of plots), on different problems (column of plots), for different GE variant (color of line).

Table 2: AEP for the crossover operator.

Problem	$\|g\|$	GE	BGE	πGE	SGE
Harmonic	75				0.129
	128	0.023	0.029	0.049	
	256	0.017	0.023	0.034	
	512	0.016	0.014	0.026	
	1024	0.01	0.01	0.016	
Polynomial	121				0.097
	128	0.027	0.033	0.058	
	256	0.019	0.02	0.036	
	512	0.009	0.012	0.02	
	1024	0.004	0.004	0.01	
Santa-Fe	31				0.058
	128	0.103	0.095	0.099	
	256	0.064	0.079	0.096	
	512	0.055	0.06	0.077	
	1024	0.044	0.058	0.077	
Text	85				0.113
	128	0.14	0.196	0.178	
	256	0.135	0.166	0.189	
	512	0.086	0.094	0.144	
	1024	0.057	0.051	0.083	

(and the parameter $d_{\max}$), no strong conclusions can be drawn for that variant about the impact of $|g|$ on AEP; however, by observing Figure 3, points for SGE seem to be roughly consistent with the shape of the curves for the other variants, hence suggesting that the value of $d_{\max}$ suggested by SGE authors might be not optimal.

From another point of view, this confirms that choosing a proper value for that parameter is not easy.

Third, the absolute values of AEP for the 4 benchmark problems appear to reflect their nature. Harmonic and Polynomial look similar, in accordance with the fact that they have very similar grammars and are both simbolic regression problems. On the other hand, Santa-Fe and Text exhibit larger values for AEP, suggesting that they are easier problems: in these terms, Text is both the problem with largest values for AEP and the one in which the differences among variants are the sharpest.

5.3 Fitness-Probability Cloud

In order to gain further insights in the escape probability, we analyzed in more detail the data for $|g| = 1024$ (or the natural genotype size for SGE). Figure 4 shows the plots of the Fitness-Probability Cloud for each combination of problem, operator, and variant. We recall that each the plot of the Fitness-Probability Cloud shows on the x-axis the fitness of the best parent (or of the only parent, for the mutation) and on the y-axis the escape probability. Since we generated 300 different parent pairs for each combination and in order to increase the plots clarity, we (i) discarded the 25% of pairs with the worst fitness values, (ii) grouped the remaining values in 10 bins of equals width, resulting in an uneven (and possibly zero) number of pairs in each bin, and finally (iii) averaged the results across tuples in each bin. The removal of the worst quartile was beneficial to clarity in particular for the Harmonic and Polynomial problems, for which the fitness is not bounded.

It can be seen from Figure 4 that the worse the fitness of the best parent, the greater the escape probability—recall that, for all our problems, the fitness has to be minimized. This finding is sound and is consistent with the typical shape of the (best) fitness vs. generation plot in which the improvements to the best individual

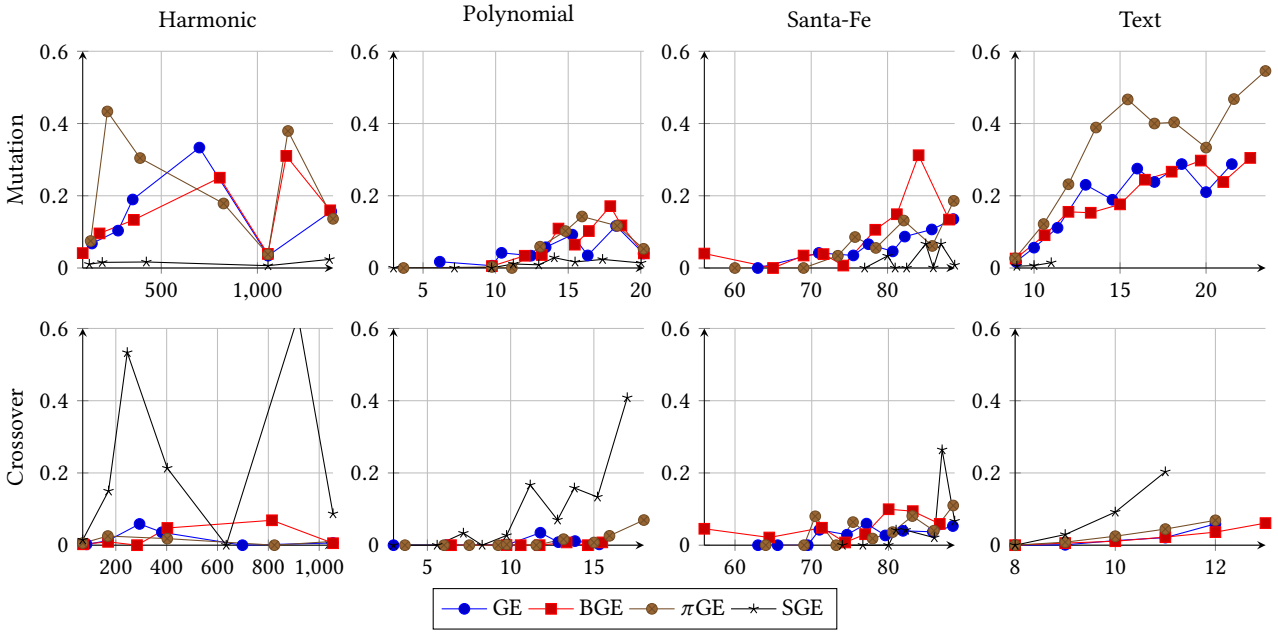

Figure 4: Fitness-Probability Clouds for genotype size $|g| = 1024$ (or natural size for SGE), with different genetic operators (row of plots), on different problems (column of plots), for different GE variant (color of line). Caveat: lower fitness is better.

are apparent at the beginning of the evolution, yet less evident in later phases, when the average individual in the population has a better fitness.

In addition to the difference between SGE and the other three variants, the Fitness-Probability Clouds of Figure 4 do not allow to spot any significant dissimilarity in the behavior of the variants, that is, the shape of the curves is roughly the same in all cases, with the exception of the Harmonic problem. We analyzed in finer detail the data for that problem and we found that the fitness values are often very large, due to the presence of the division operator in the grammar, resulting in very sparse data on the x-axis. However, the leftmost part of the curve (that is, the one corresponding to better fitness) for both mutation and crossover is consistent with the curves of the other problems.

5.4 Explaining evolvability

We tried to identify which factors can better explain the evolvability of GE. We focused on locality and redundancy, two properties of GE which measure how it complies with the general principle of *variational inheritance* [5] and which have been widely studied [3, 14, 18, 22, 23]. From an high level point of view, *locality* in GE measures to which degree small modifications to the genotype of an individual result in small modifications in the corresponding phenotype; *redundancy* in GE measures how often different genotypes are mapped to the same phenotype.

For precisely quantifying locality and redundancy, we adopted the framework of [14], according to which the locality is the Pearson correlation between the shortest parent-child genotype distance $\min(d_g(g_p^1, g_c), d_g(g_p^2, g_c))$ and the shortest parent-child phenotype distance $\min(d_p(p_p^1, p_c), d_p(p_p^2, p_c))$; redundancy is the percentage

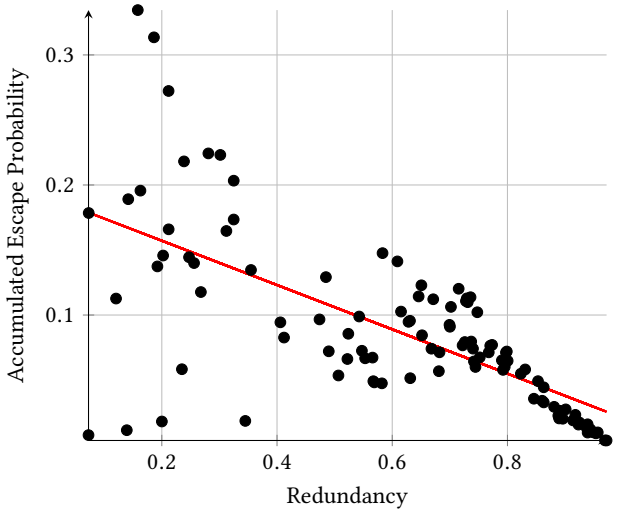

Figure 5: AEP vs. redundancy: each point represents a combination of problem, variant, operator, and genotype size; the red line is a regression line.

of cases in which the child genotypes is different from both parent genotypes (that is, $\min(d_g(g_p^1, g_c), d_g(g_p^2, g_c) > 0$) and the child phenotype is equal to at least one parent phenotype (i.e., $\min(d_p(p_p^1, p_c), d_p(p_p^2, p_c)) = 0$). Both indexes are computed over a sample of genetic operator applications. As in the cited paper, we used the Hamming distance for the genotype (that is, d_g) and

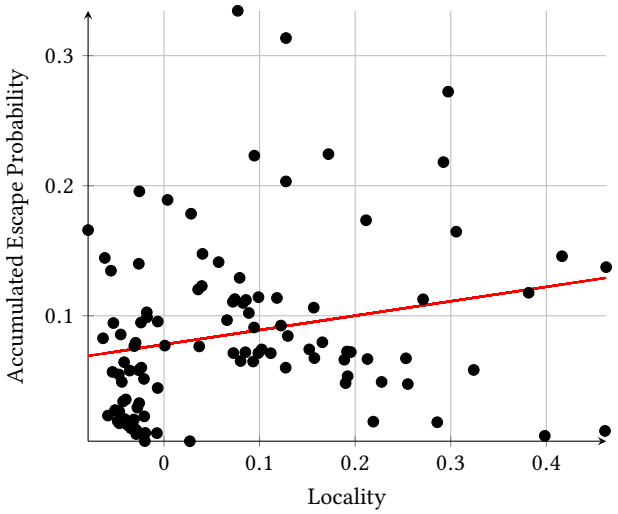

Figure 6: AEP vs. locality: each point represents a combination of problem, variant, operator, and genotype size; the red line is a regression line.

the edit distance between strings of the language $\mathcal{L}(\mathcal{G})$ for the phenotype (that is, d_p).

We measured locality and redundancy separately for each combination of problem, variant, operator, and genotype size, i.e., on the very same data on which we computed AEP. Figures 5 and 6 show the results: each figure plots a point for each combination with the AEP on the y-axis and the redundancy (Figure 5) or locality (Figure-6) on the x-axis, along with a linear regression line. We recall that all the three indexes have a limited domain, regardless of the mapper, the problem, and the genetic operator—$[0, 1]$ for AEP and redundancy and $[-1, 1]$ for locality. Thus, they can be meaningfully compared in different conditions.

Figure 5 suggests that there is a clear influence of the redundancy on the evolvability, the latter being measured with AEP: the greater the redundancy, the lower the evolvability. Since redundancy operates on a different level (genotype-phenotype mapping) than AEP (fitness computation), it is fair to claim that redundancy has a causal effect on the evolvability, which is particularly strong for greater values of redundancy. We explain this finding as follows: if the genotype-phenotype mapping struggles in generating a phenotype which is different from both the parents, the corresponding fitness cannot be better (and neither worse) than the one of the parents. In this terms, we speculate that any approach addressing this limitation (e.g., a less redundant mapping function or some diversity promotion mechanism in which clones are discarded or disadvantaged) may be beneficial to the evolvability and, eventually, to the overall effectiveness of the EA.

Figure 6, on the other hand, does not highlight any clear relation between the locality and AEP: the coefficient of determination for the corresponding linear regression model is low ($R^2 = 0.05$ vs. $R^2 = 0.45$ for redundancy). We explain this finding as follows: locality measures how the strength of a modification on the genotype is proportional to the strength of the corresponding modification on the phenotype; however, provided that some modification has occurred, no guarantees exist that it will positively affect the fitness. From a general point of view, our finding might be an experimental explanation for the claims of some previous works in which the authors questioned if the locality in GE is indeed beneficial to its effectiveness [8].

6 CONCLUDING REMARKS

The genotype-phenotype mapping function of GE plays a crucial role and can be considered the main motivation for the wide adoption of GE in many different application domains. On the other hand, several studies scrutinized the GE mapping function and found that it scarcely adheres to the variational inheritance principle, exhibiting low locality and high redundancy.

In this paper, we experimentally studied the evolvability of GE in different conditions (problem, mapping function, genotype size, and genetic operator) using a method recently proposed for analyzing this property, namely consisting of the Accumulated Escape Probability and the Fitness-Probability Cloud. To the best of our knowledge, this is the first study of this kind for GE.

Our findings based on the experimental results are twofold. On the one hand, none of the aforementioned factors, taken in isolation, strongly affects the evolvability: in particular, no one of the 4 mapping functions that we considered performs well with both the mutation and the crossover operators. On the other hand, a deeper analysis shows that redundancy has a clear impact on evolvability, much larger than the impact of locality. We believe that our findings may constitute a foundation for future research about GE and may stimulate researchers and practitioners in designing methods for addressing GE intrinsic redundancy.

Acknowledgements

F.D. is supported by the EPSRC [grant number EP/J017515/1].

REFERENCES

[1] Lee Altenberg and others. 1994. The evolution of evolvability in genetic programming. *Advances in genetic programming* 3 (1994), 47–74.
[2] Alberto Bartoli, Andrea De Lorenzo, Eric Medvet, and Fabiano Tarlao. 2016. Syntactical Similarity Learning by Means of Grammatical Evolution. In *Parallel Problem Solving from Nature – PPSN XIV: 14th International Conference, Edinburgh, UK, September 17-21, 2016, Proceedings*. Springer International Publishing, Cham, 260–269. DOI: http://dx.doi.org/10.1007/978-3-319-45823-6_24
[3] Tom Castle and Colin G Johnson. 2010. Positional effect of crossover and mutation in grammatical evolution. In *European Conference on Genetic Programming*. Springer, 26–37.
[4] Joseph C Culberson. 1998. On the futility of blind search: An algorithmic view of "no free lunch". *Evolutionary Computation* 6, 2 (1998), 109–127.
[5] Kenneth A De Jong. 2006. *Evolutionary computation: a unified approach*. MIT press.
[6] David Fagan, Michael O'Neill, Edgar Galván-López, Anthony Brabazon, and Sean McGarraghy. 2010. An analysis of genotype-phenotype maps in grammatical evolution. In *European Conference on Genetic Programming*. Springer, 62–73.
[7] Christopher J Headleand, Llyr Ap Cenydd, and William J Teahan. 2014. Benchmarking Grammar-Based Genetic Programming Algorithms. In *Research and Development in Intelligent Systems XXXI*. Springer, 135–148.
[8] Jonatan Hugosson, Erik Hemberg, Anthony Brabazon, and Michael O'Neill. 2007. An investigation of the mutation operator using different representations in grammatical evolution. In *Proc. 2nd International Symposium Advances in Artificial Intelligence and Applications*, Vol. 2. 409–419.
[9] Maarten Keijzer, Michael O'Neill, Conor Ryan, and Mike Cattolico. 2002. Grammatical evolution rules: The mod and the bucket rule. In *European Conference on Genetic Programming*. Springer, 123–130.
[10] Nuno Lourenço, Francisco B Pereira, and Ernesto Costa. 2015. SGE: a structured representation for grammatical evolution. In *International Conference on Artificial Evolution (Evolution Artificielle)*. Springer, 136–148.

[11] Nuno Lourenço, Francisco B Pereira, and Ernesto Costa. 2016. Unveiling the properties of structured grammatical evolution. *Genetic Programming and Evolvable Machines* (2016), 251–289.

[12] Guanzhou Lu, Jinlong Li, and Xin Yao. 2011. Fitness-probability cloud and a measure of problem hardness for evolutionary algorithms. In *European Conference on Evolutionary Computation in Combinatorial Optimization*. Springer, 108–117.

[13] Katherine M Malan and Andries P Engelbrecht. 2013. A survey of techniques for characterising fitness landscapes and some possible ways forward. *Information Sciences* 241 (2013), 148–163.

[14] Eric Medvet. 2017. A Comparative Analysis of Dynamic Locality and Redundancy in Grammatical Evolution. In *Genetic Programming: 20th European Conference, EuroGP 2017, Amsterdam, Netherlands, April 19-21, 2017, Proceedings*. Springer International Publishing, Cham, to appear.

[15] Michael O'Neill, Conor Ryan, Maarten Keijzer, and Mike Cattolico. 2003. Crossover in grammatical evolution. *Genetic programming and evolvable machines* 4, 1 (2003), 67–93.

[16] Michael O'Neill, Anthony Brabazon, Miguel Nicolau, Sean Mc Garraghy, and Peter Keenan. 2004. πgrammatical evolution. In *Genetic and Evolutionary Computation Conference*. Springer, 617–629.

[17] John O'Sullivan and Conor Ryan. 2002. An investigation into the use of different search strategies with grammatical evolution. In *European Conference on Genetic Programming*. Springer, 268–277.

[18] Franz Rothlauf and Marie Oetzel. 2006. On the locality of grammatical evolution. In *European Conference on Genetic Programming*. Springer, 320–330.

[19] Conor Ryan, Atif Azad, Alan Sheahan, and Michael O'Neill. 2002. No coercion and no prohibition, a position independent encoding scheme for evolutionary algorithms–the chorus system. In *European Conference on Genetic Programming*. Springer, 131–141.

[20] Conor Ryan, JJ Collins, and Michael O'Neill. 1998. Grammatical evolution: Evolving programs for an arbitrary language. In *European Conference on Genetic Programming*. Springer, 83–96.

[21] Ann Thorhauer. 2016. On the Non-uniform Redundancy in Grammatical Evolution. In *International Conference on Parallel Problem Solving from Nature*. Springer, 292–302.

[22] Ann Thorhauer and Franz Rothlauf. 2014. On the locality of standard search operators in grammatical evolution. In *International Conference on Parallel Problem Solving from Nature*. Springer, 465–475.

[23] Nguyen Quang Uy, Nguyen Xuan Hoai, Michael O'Neill, R.I. McKay, and Dao Ngoc Phong. 2013. On the roles of semantic locality of crossover in genetic programming. *Information Sciences* 235 (jun 2013), 195–213. DOI: http://dx.doi.org/10.1016/j.ins.2013.02.008

[24] Sébastien Verel, Philippe Collard, and Manuel Clergue. 2003. Where are bottlenecks in nk fitness landscapes?. In *Evolutionary Computation, 2003. CEC'03. The 2003 Congress on*, Vol. 1. IEEE, 273–280.

[25] David R White, James Mcdermott, Mauro Castelli, Luca Manzoni, Brian W Goldman, Gabriel Kronberger, Wojciech Jaśkowski, Una-May O'Reilly, and Sean Luke. 2013. Better GP benchmarks: community survey results and proposals. *Genetic Programming and Evolvable Machines* 14, 1 (2013), 3–29.