Multi-level diversity promotion strategies for Grammar-guided Genetic Programming

Multi-level Diversity Promotion Strategies for Grammar-guided Genetic Programming

Alberto Bartoli[a], Andrea De Lorenzo[a], Eric Medvet[a], Giovanni Squillero[b]

[a] *DIA, University of Trieste, Trieste, Italy*
[b] *Politecnico di Torino, Torino, Italy*

Abstract

Grammar-guided Genetic Programming (G3P) is a family of Evolutionary Algorithms that can evolve programs in any language described by a context-free grammar. The most widespread members of this family are based on an indirect representation: a sequence of bits or integers (the genotype) is transformed into a string of the language (the phenotype) by means of a mapping function, and eventually into a fitness value. Unfortunately, the flexibility brought by this mapping is also likely to introduce non-locality phenomena, reduce diversity, and hamper the effectiveness of the algorithm. In this paper, we experimentally characterize how population diversity, measured at different levels, varies for four popular G3P approaches. We then propose two strategies for promoting diversity which are general, independent both from the specific problem being tackled and from the other components of the Evolutionary Algorithm, such as genotype-phenotype mapping, selection criteria, and genetic operators. We experimentally demonstrate their efficacy in a wide range of conditions and from different points of view. The results also confirm the preponderant importance of the phenotype-level analyses in diversity promotion.

Keywords: Representation, Grammatical Evolution, CFGGP, SGE, WHGE

1. Introduction

Grammar-guided Genetic Programming (G3P) may be considered as a natural extension of the original paradigm introduced by Koza in late 1980s [1]. Differently from Genetic Programming (GP), G3P exploits a grammar in order to ensure that all the individuals in the population are syntactically valid.

While the embryonic idea of using a grammar may be attributed to Koza himself [2], the first line of research that can be sensibly labeled "grammar-guided" dates back to mid-1990s, with Whigham's Context-free Grammar Genetic Programming (CFGGP) [3] and Geyer-Schulz's rule-based expert system [4]. Here, phenotypes are still trees, but are derived according to an arbitrary *context-free grammar* and genetic operators are designed to preserve this representation.

Grammatical Evolution (GE), probably the best known G3P approach, has been proposed by Ryan, Collins, and O'Neill in 1998 [5]. It encodes individuals into genomes as unstructured, variable-length sequences of bits grouped in *codons*, eventually interpreted in the context of a user-supplied grammar. More specifically, the integer values of the codons are used to select among the list of possible derivations in a grammar in the Backus-Naur form. This procedure is a *mapping* from the individual represented as a bit string to the resulting string of the language defined by the grammar—GE is thus said to adopt an *indirect* representation of the individuals.

The main advantage of G3P is apparent: changing the base grammar allows to exploit the very same Evolutionary Algorithm (EA) for virtually any possible problem without modification. On the other hand, the mapping procedure which characterizes GE has been shown to impair the evolution process [6]. The *locality* of a representation describes how much small genotypic changes caused by the application of the genetic operators correspond to small changes in the fitness of individuals. It has been widely acknowledged by scholars that "high-locality representations preserve the difficulty of a problem and phenotypically easy problems also remain genotypically easy. Using low-locality representations is equivalent to randomizing the search process." [7]. GE may exhibit a remarkable low-locality, as the change of a single bit in the genome is likely to affect many different derivations and, eventually, to result in a largely different fitness.

In the past 25 years literature reported several successful application of GE [8], together with scholarly articles that scrutinize its peculiar evolutionary processes [9, 10, 11, 6]. Among these, a few proposals arose for a different mapping which could address the limitations of the original GE mapping, e.g., πGE [12], SGE [13], WHGE [14]. A crucial problem that emerged from such studies is that mapping

Email addresses: bartoli.alberto@units.it (Alberto Bartoli), andrea.delorenzo@units.it (Andrea De Lorenzo), emedvet@units.it (Eric Medvet), giovanni.squillero@polito.it (Giovanni Squillero)

may impact on the population *diversity*: in particular, the tendency to map different genotypes to the same phenotype may result in many individuals being the same and, eventually, may lead to premature convergence [15, 6, 16]. While the lack of diversity is not necessarily a problem *per se*, it is frequently associated with poor performances. Diversity is not the end goal of an EA, but promoting it can be an important mean goal.

In this paper, we address this topic in depth. We first analyze experimentally four G3P approaches (CFGGP, GE, SGE, and WHGE—see Section 3) in order to understand if and how they are affected by lack of diversity. Then, we propose two general strategies for promoting diversity in G3P, one being an adaptation of an existing diversity promotion strategy—namely deterministic crowding [17]—to G3P. Both strategies are independent from the problem tackled and the details of the fitness function. Not being dependent on the structure of the solution nor on the actual grammar, the strategies are independent from the genotype-phenotype mapping. Moreover, the two diversity promotion strategies are not influenced by the characteristics of the EA, such as the selection criteria or the genetic operators. They may be set to operate at a very specific level, namely, genotype, phenotype, or fitness. Beyond the goal of improving G3P effectiveness, and hence further extend its applicability, our study aims at better understanding how diversity promotion may impact on EAs based on indirect representations.

We performed a thorough experimental analysis based on 8 benchmark problems and 4 G3P variants, differing in the representation of the individuals. We show that the considered G3P variants indeed have an issue of lack of diversity and we also show that diversity promotion always results in an improvement of the search effectiveness: regardless the G3P variant being used and the problem being tackled, some of the diversity promotion strategies here considered always lead to a better final best fitness (on average). The experimental results suggest that similar mechanisms could be beneficial for different EAs.

A brief and preliminary study along the same line of this paper has been presented in [18]. Here, we extend the cited paper in several ways: (a) we provide a much deeper discussion of the diversity promotion strategy proposed in [18] and consider another strategy based on the adaptation of deterministic crowding to G3P; (b) we apply the two strategies to 4 variants of G3P (CFGGP, GE, SGE, WHGE), instead of only on GE; (c) we perform a much deeper experimental evaluation considering a larger set of benchmark problems and analyzing the results in greater detail.

The remainder of the article is organized as follows. In Section 2, we survey the relevant literature with respect to diversity promotion. In Section 3, we give a common formulation of G3P techniques and then describe in details the 4 different considered G3P variants. In Section 4, we introduce the two strategies for diversity promotion in G3P. In Section 5, we describe the experimental evaluation and discuss the results. Finally, in Section 6, we draw the conclusions.

2. Related works

The *lack of diversity* frequently limits the effectiveness of evolutionary algorithms: Holland himself analyzed the issue, talking about the *lack of speciation* in his seminal works [19]. It is, however, an endemic phenomenon, possibly rooted in the very use of a *fitness function* instead of a real environment [15]. A set of diverse individuals is not the final goal, yet, most scholars agree that enforcing a higher level of diversity within the population may be beneficial for the overall evolutionary process.

The *lack of diversity* is not common in nature. On the contrary, Darwin called the "divergence of character", exactly the opposite phenomenon, a cornerstone of his theory—variations increasing diversity are likely to be favored as the more the co-inhabitants of an area differ in their ecological requirements, the less they will compete [20]. Inspired by this reasoning, scholars tried to artificially limit the competition between individuals, artificially partitioning the environment to increase the global diversity. The resulting *niching methods* have been demonstrated among the most effective mechanisms to promote diversity in evolutionary computation [17, 21].

Several type of niching have been proposed in literature, some under different names. The common element is that the opportunities to generate offspring for an individual are influenced by the number of other individuals occupying the same "niche": the more a spot in the search space is crowded, the less chances its occupants get. But, apart from the core idea, details differ greatly. Niching methods are usually divided into two broad classes: *explicit neighborhood* methods, that require an explicit definition of the size of a niche through a parameter called *niche radius*; and *implicit neighborhood* methods, where the algorithm requires no information about the search space. The former are applicable whenever the difference between individuals can be sensibly measured: all solutions differing less than a given threshold from the current one are considered part of the same niche. The latter may be exploited whether the similarity can be directly inferred, for instance the offspring will occupy the same niche of the parents.

A crucial component of any explicit neighborhood method is the *distance metric* used, which is necessary for quantifying the difference between individuals and determining whether two individuals belong to the same niche. In this respect, most methods work at the level of genotypes as at this level it is typically easy to define a distance metric, e.g., Hamming distance for bit strings. On the other hand, such methods are effective only when the distance between genotypes is related to the distance between phenotype (i.e., when the locality principle is satisfied) but such a relation is often very feeble.

Many application domains have benefited from distance definitions able to capture an application-specific similarity criterion more effectively than a generic distance definition [22, 23, 24] while other domains have instead benefited of very general distance definitions rooted on Kolmogorov complexity [25, 26]. Approaches of this kind are unlikely to be useful in the GE framework because of the very same nature of the genotype-phenotype GE mapping procedure, that often exhibits a tendency of mapping multiple different genotypes on the same phenotype. This important phenomenon is called *degeneracy*, and it has been scrutinized since the very start of this research line [27, 28, 29, 30, 31, 32, 33, 34, 35]. Degeneracy makes even more apparent the difference between diversity at the two levels [36]. Mutations may be *silent*, or *neutral*, as they may change the genotype without affecting the phenotype; and parents that exhibit the same phenotype could produce different offspring, as they might not share the same genotype.

Diversity may be increased also by reducing the selective pressure, i.e., by acting on the exploration-exploitation trade-off which is common to all population-based EAs [37]. A well known way of reducing or increasing the selective pressure is to tune or modify the selection criteria [38]. We here focus more on the interaction between diversity and the representation: diversity promotion strategies based on different EA components, such as the selection criteria, might indeed be complementary to the one we propose.

Finally, the increase of diversity may be a side-effect of modifications on EA components which pursue different goals. In [39] GOMEA, a model-based EA which employs a particular genetic operator without a selection phase, is applied to GE, SGE, and WHGE: the authors obtain good results and report that the improvement might be explained also by the increase in diversity. In [40], it is shown that different population initialization procedures in GE may result in different degrees of diversity, measured, as in our work, as the fraction of unique phenotypes. Similarly, in [41] an experimental analysis is presented of the impact of different grammar-related design choices (i.e., not related to the EA itself) on several aspects of GE, including diversity.

3. G3P

In this section, we describe the G3P variants analyzed in the experimental assessment. We considered four variants: Context-free grammar Genetic Programming (CFGGP) [42], standard Grammatical Evolution (GE) [5], Structured Grammatical Evolution (SGE) [13, 43], and Weighted Hierarchical Grammatical Evolution (WHGE) [44, 14]. We describe these proposals in terms of a common framework, presented in Section 3.1, where a fixed-size population is evolved iteratively.

All the variants are based on an *indirect* representation, i.e., individuals are represented by a genotype and a phenotype, the latter being obtained from the former by means of a strategy-specific *mapping function*[1]. Performances in real-world applications are greatly influenced by the type of representation adopted, but the choice of a specific encoding often arises from intuition and guesswork as proper guidelines are not available. Our choice of the 4 variants is arbitrary: however, we attempted to include approaches with different base representations (bit and integer strings, trees) and different mapping properties [44]. In particular, we chose the original GE which uses bit strings, instead of its later adaptations which use integer strings, because bit strings—used since the seminal works of EC—are still the encoding favored by scholars interested in the algorithms themselves, in evolution dynamics, and in the phenotype-genotype mappings [45].

The 4 variants are described in detail in sections 3.2–3.5.

3.1. Steady-State G3P

The *Steady-State G3P* (SSG3P) [46, 47] evolves a fixed size population of n_{pop} individuals initially built according to an initialization procedure InitPopulation(). The population is evolved by iteratively applying one genetic operator to one (mutation) or two (crossover) parent individuals: the resulting offspring is added to the population and, in order to keep the population size fixed, a corresponding number of individuals are then removed. In details, the evolution occurs according to the following iterative procedure (also described in Algorithm 1):

1. A genetic operator is selected randomly between crossover and mutation (GetOperator() function) with probability $p_{\text{op,cross}}$ and $1-p_{\text{op,cross}}$, respectively.

2. The proper number of parents with respect to the selected genetic operator are selected from the population (*parent selection* function SelPar()).

3. The genetic operator is applied to the parent genotypes, resulting in one or more new children genotypes (Apply()) function): the children genotypes are then mapped to phenotypes according to a *genotype-phenotype mapping* (Map() function) and their fitness is computed (Fitness() function).

4. The offspring are added to the population.

5. Until the population size is greater than n_{pop}, the following steps are repeated: (i) one individual is selected from the population according to a *removal selection* function SelRemoval() and (ii) the selected individual is then removed from the population.

The procedure is repeated for a fixed number of times, i.e., a fixed number of individuals are born by applying

[1] Although CFGGP does not explicitly use an indirect representation, it does fit our general treatment by considering the mapping function as the identity function

the genetic operators. Conventionally, the termination criterion is expressed in terms of a predefined number n_{gen} of generations, a generation consisting of n_{pop} births—the procedure is hence iterated n_{gen} times, until $n_{\text{gen}}n_{\text{pop}}$ individuals have been generated.

Algorithm 1 SSG3P evolution algorithm.

```
procedure EVOLVE()
    I ← INITPOPULATION(n_pop)
    n ← 0
    while n < n_gen n_pop do
        o ← GETOPERATOR(p_op,cross)
        G_p ← ∅
        while |G_p| < ARITY(o) do
            (g_p, p_p, f_p) ← SELPAR(I)
            G_p ← G_p ∪ {g_p}
        end while
        G_c ← APPLY(o, G_p)
        for all g_c ∈ G_c do
            p_c ← MAP(g_c)
            f_c ← FITNESS(p_c)
            I ← I ∪ {(g_c, p_c, f_c)}
        end for
        while |I| > n_pop do
            I ← I \ {SELREMOVAL(I)}
        end while
        n ← n + |G_c|
    end while
end procedure
```

The procedure described above is agnostic to the specific functions invoked for population initialization (INITPOPULATION()), application of a genetic operator (APPLY()), and so on. A common choice for selecting the parents (SELPAR())) is *tournament selection* while a common choice for selecting the individual to remove from the population (SELREMOVAL()) is worst fitness (i.e., *truncation selection*). Concerning the genotype-phenotype mapping function MAP(), in this study we considered 4 among the most significant and recent variants that we describe in the following sections.

In all the variants, the phenotype is a string of the language $\mathcal{L}(\mathcal{G})$ defined by a context-free grammar (CFG) $\mathcal{G} = (N, T, s_0, R)$, where N is the set of non-terminal symbols, T is the set of terminal symbols (with $T \cap N = \emptyset$), $s_0 \in N$ is the starting symbol, and R is the set of derivation rules. Each derivation rule describes how a non-terminal symbol of N may be replaced by a sequence of symbols of $N \cup T$: the application of a rule, i.e., the actual replacement operation, is called *derivation*. The subset of rules for a symbol s is denoted by R_s.

Figure 1 shows an example CFG using the Backus-Naur Form (BNF): this notation specifies, using a common convention, all the relevant information about the grammar. Each line specifies all the derivation rules for a given non-terminal, shown before ::= . Then, the possible derivations are separated by | . The starting symbol $s_0 = \langle \text{expr} \rangle$ is the non-terminal symbol on the left-side of the first line. The set $R_{\langle \text{expr} \rangle} \subset R$ of derivation rules for the non-terminal $\langle \text{expr} \rangle$ are: $\langle \text{expr} \rangle \rightarrow (\ \langle \text{expr} \rangle\ \langle \text{op} \rangle\ \langle \text{expr} \rangle\ )$, $\langle \text{expr} \rangle \rightarrow \langle \text{num} \rangle$, and $\langle \text{expr} \rangle \rightarrow \langle \text{var} \rangle$. Similarly, the BNF of Figure 1 specifies the subsets $R_{\langle \text{op} \rangle}$, $R_{\langle \text{var} \rangle}$, and $R_{\langle \text{num} \rangle}$.

```
⟨expr⟩ ::= ( ⟨expr⟩ ⟨op⟩ ⟨expr⟩ ) | ⟨num⟩ | ⟨var⟩
⟨op⟩ ::= + | - | * | /
⟨var⟩ ::= x | y
⟨num⟩ ::= 0.1 | 1 | 10
```

Figure 1: A CFG in the Backus-Naur Form (BNF) for mathematical expressions.

3.2. Context-free grammar Genetic Programming

CFGGP [42] is one of the first approaches for G3P, and it is still quite popular. In CFGGP the genotype g is a *derivation tree* consistent with the grammar $\mathcal{G}$ in which leaf nodes are terminals $s \in T$, non-leaf nodes are non-terminals $s \in N$, the root is the starting symbol s_0, and a node is child of another node if the former has been inserted in the phenotype upon the derivation of the latter.

The MAP() function simply consists in concatenating the leaf-nodes of g from the left to the right, obtaining hence a string $p \in \mathcal{L}(\mathcal{G})$.

Concerning the initialization procedure (INITPOPULATION()), we considered the *Ramped Half-and-Half* method [48] in which half of the genotypes are built with the "grow" method and half with the "full" method. Both methods generate derivation trees randomly—by taking into account the CFG—and use a parameter d: with the "grow" method, the tree g is such that at least one leaf-node is at depth d, whereas other leaf-nodes are at depth lower than or equal to d; with the "full" method, all leaf-nodes are at depth d.

Concerning the application of genetic operators (APPLY()), the genetic operators of CFGGP must ensure that their application results in valid derivation trees. The mutation operator consists in (i) randomly choosing a non-leaf node corresponding to a non-terminal $s \in N$ and (ii) replacing the subtree starting in the chosen node with a tree built with the "grow" method and a maximum subtree depth of $d - d_s$, where d_s is the depth of the replaced node. The crossover operator consists in (i) randomly choosing two non-leaf nodes in the two parent trees such that they correspond to the same non-terminal $s \in N$ and (ii) exchanging the corresponding subtrees. The second step is performed only if both the resulting trees have (after the exchange) a depth lower than or equal to d, otherwise, the first step is repeated.

3.3. Standard Grammatical Evolution

In standard GE [5], the genotype is a bit string which is viewed as a string of integers obtained by decoding substrings of n (usually set to $n = 8$) consecutive bits: each integer is called *codon*.

The MAP() function for GE consists in repeating the following steps, starting with the phenotype $p = s_0$, a counter $i = 0$, and a counter $w = 0$.

1. Derive the leftmost non-terminal s in p using the j-th derivation rule (zero-based indexing) in $R_s \subset R$, where $j = g_i \mod |r_s|$, i.e., 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.

2. Increment i and compare it against the number of codons $\frac{|g|}{n}$: if $i > \frac{|g|}{n}$, set $i = 0$ and increment w. If w is greater than a predefined threshold n_w, then abort the procedure and return a *null phenotype*.

3. If p contains at least one non-terminal, return to step 1, otherwise end.

The above steps are repeated until p does not contain any non-terminal symbol.

The re-use of the genotype, triggered by the first condition at step 2 and required to cope with recursive CFGs, that is, with infinite languages, is called *wrapping*; a maximum of n_w wrappings are allowed. If n_w wrappings are executed and the phenotype p still contains non-terminal symbols, MAP(g) returns a null (also known as invalid) genotype which is then conventionally associated with the worst possible fitness value.

GE may work together with any genetic operator suitable for bit strings: common choices are the bit flip mutation and the two-points crossover.

3.4. Structured Grammatical Evolution

In the recently proposed SGE [13, 43] the genotype consists of fixed-size lists (*genes*) of integers: each list corresponds to a non-terminal symbol and each integer in the list determines a single derivation of that non-terminal. SGE lacks a mechanism for reusing the genotype: instead, the ability of coping with infinite languages is obtained by working with a non-recursive grammar $\mathcal{G}'$ derived automatically from the input grammar $\mathcal{G}$. For the latter transformation, the user must provide a value for the parameter $d_{\max}$ which represents the maximum level of recursion of derivation rules.

In detail, the genotype g is composed of $|N|$ genes and each gene g_s, with $s \in N$ a non-terminal of $\mathcal{G}$, contains a number of codons determined by the grammar and $d_{\max}$: each codon assumes values in $\{0, \ldots, |R_s| - 1\}$, $|R_s|$ being the number of derivation rules for s.

The MAP() function for SGE consists in repeating the following steps, starting with the phenotype $p = s_0$ and a counter $i_s = 0$ for each non-terminal $s \in N$.

1. Expand the leftmost non-terminal s in p by using the g_{s,i_s}-th derivation rule (zero-based indexing) in R_s, with g_{s,i_s} denoting the value of the i_s-th codon (zero-based indexing) in g_s.

2. Increment i_s.

The above steps are repeated until p does not contain any non-terminal symbol; MAP() always returns a non-null phenotype.

Differently than GE, SGE relies on genetic operators which are tailored to the specific SGE representation. The mutation operator consists in, for each codon, changing its value to a new random value in the appropriate domain, which differs from gene to gene, with a probability p_{mut}. The crossover operator exchanges the genes g_s^1, g_s^2 of the parent genotypes, for each non-terminal s in a randomly chosen subset $N' \subseteq N$: i.e., exchange of genetic material between the parents occurs for either all or none of the codons in a gene. SGE mutation and crossover resemble classic mutation and crossover operators: we remark, however, that they do take into account the peculiar SGE representation.

The population initialization procedure consists in setting random values, chosen with uniform probability in the appropriate domain, for each codon.

3.5. Weighted Hierarchical Grammatical Evolution

WHGE [44, 14] is the most recent variant of G3P. In WHGE, as in GE, the genotype is an unstructured bit string: as a consequence, WHGE does not impose any constraint on the population initialization procedure, nor on the genetic operators (i.e., any genetic operator which works with bit strings may be used with WHGE).

The MAP() function of WHGE operates in two phases: in a first phase, the genotype g is transformed in a derivation tree by means of a recursive mapping function which we here denote with REMAP(); in a second phase, the derivation tree is transformed in the phenotype $p \in \mathcal{L}(\mathcal{G})$ by concatenating the leaf-nodes of the derivation tree from the left to the right.

The recursive mapping function REMAP(s, g') takes as arguments a symbol $s \in N \cup T$ and a bit string g' and returns a derivation tree. The function is first called with arguments being the starting symbol s_0 and the genotype g: if $s \in T$, MAP(s, g') returns a tree composed of the only symbol s. Otherwise, the function consists in the following steps.

1. If $|g'| \geq |r_s|$, then: (i) split g' in $|r_s|$ substrings of equal length or, if not possible, in a way such that the variance of the lengths is the lowest; (ii) find the index i for which the *relative cardinality* (i.e., ratio between the count of bits set to 1 and the number of all bits) of the i-th substring of g' is the largest or, in case of tie, the lowest index among ties. Otherwise, i.e., if $|g'| < |r_s|$, choose the i-th derivation rule in R_s which leads to a sequence of terminals in the lowest number of derivations starting from s.

2. Split g' in n non-overlapping portions whose length are proportional to $e_{s_1}, \ldots, e_{s_n}$, where $s_1, \ldots, s_n$ are the symbols resulting from the derivation of s according to the i-th derivation rule in R_s and e_{s_j} is the expressive power of the symbol s_j (see below).

3. Build a tree t with s as root and, as children, the trees returned by calling ReMap(s_j, g'_j) for each symbol s_j and corresponding substring g'_j obtained at the previous step.

4. Return the tree t.

The expressive power e_s is computed just once for each non-terminal $s \in N$ at the beginning of the evolution. It corresponds to the number of different derivation trees which can be built with s as root: however, since e_s could be infinite with recursive grammars, a maximum depth n_d is imposed while computing e_s, n_d being a parameter of WHGE.

Similarly to CFGGP and SGE, in WHGE Map() always returns a non-null phenotype.

4. Strategies for promoting diversity

We here describe the two strategies for diversity promotion: Partitioned Population G3P (PPG3P) and Deterministic Crowding G3P (DCG3P). Both are multi-level, in the sense that they can be configured to work at the level of the genotype, phenotype, or fitness. The former has been presented in [18], the latter is based on the original idea *deterministic crowding* [17], but adapted to the case of G3P.

4.1. Partitioned Population G3P

PPG3P, the first proposed enhancement, requires an *equivalence relation* between pairs of individuals (Eq() function in the description below). We explored 3 different options consisting in considering two individuals equivalent when they have: the same genotype, the same phenotype, the same fitness value. PPG3P partitions the population in sets of individuals according to the chosen equivalence relation. We call such sets *niches*. Whenever an individual has to be selected for reproduction, first a *representative* individual is selected from each niche (function ParRep() below) and then another selection criterion is applied on the resulting set of representatives (SelPar(), like in the SSG3P strategy presented in Section 3.1). Similarly, whenever an individual has to be removed from the population, first a representative individual is selected from each niche (function RemovalRep() below) and then another selection criterion is applied on the resulting set of representatives (SelRemoval(), like in SSG3P).

In detail, the evolution in PPG3P occurs according to the following iterative procedure (also described in Algorithm 2). Initially, a set of n_{pop} individuals is built and partitioned in a set $\mathcal{I}$ of sets of individuals, according to the chosen equivalence relation Eq()—i.e., all individuals in the same set are equivalent among themselves and they are not equivalent to any other individual in other sets. Then, the following steps are iterated.

1. The genetic operator is selected randomly between crossover and mutation (GetOperator() function) with probability $p_{\text{op,cross}}$ and $1-p_{\text{op,cross}}$, respectively.

2. The proper number of parents with respect to the selected genetic operator are selected, each one as follows:

   (a) the set I' of representatives is built by selecting one individual from each partition in $\mathcal{I}$, according to a *parent representative criterion* ParRep();

   (b) the parent is selected in I' according to the parent selection criterion SelPar().

3. The genetic operator is applied to the parent genotypes, resulting in one or more new children genotypes: the children genotypes are then mapped to phenotypes according to the Map() function and their fitness is computed using the Fitness() function.

4. Each one of the new individuals is added to the proper partition I_c in $\mathcal{I}$ by using the equivalence relation Eq()—possibly, I_c is a new partition, containing only the new individual, and is added to $\mathcal{I}$. If the size $|I_c|$ of the modified partition is larger than a predefined value n_{part}, the exceeding individuals are removed according to a *representative removal criterion* RemovalRep().

5. If the number $|\mathcal{I}|$ of partitions is larger than n_{pop}, one partition is removed as follows:

   (a) the set I' of representatives is built by selecting one individual from each partition in $\mathcal{I}$, according to the representative removal criterion RemovalRep();

   (b) an individual i is selected in I' according to the removal selection criterion SelRemoval();

   (c) the partition I containing i is removed from $\mathcal{I}$.

As for SSG3P, the iterative procedure of PPG3P is repeated until $n_{\text{gen}} n_{\text{pop}}$ individuals have been generated.

It can be seen that the overall number $\sum_{I \in \mathcal{I}} |I|$ of individuals is at most $n_{\text{part}} n_{\text{pop}}$—i.e., PPG3P may maintain a population larger than SSG3P. However, in both cases exactly $n_{\text{gen}} n_{\text{pop}}$ births occur during the evolution: hence PPG3P has no advantage over SSGP in these terms.

PPG3P requires a parameter n_{part} representing the maximum size allowed for each niche. With $n_{\text{part}} = 1$, PPG3P corresponds to a simple enforcement of diversity in the population: i.e., no individuals can exist in the population such that they are equivalent according to Eq(). In that case, the ParRep() and RemovalRep(c)riteria play no role in the evolution.

The other (functional) parameters of PPG3P are Eq(), ParRep() and RemovalRep(). As observed above, we explored 3 options for Eq() consisting in considering two

Algorithm 2 PPG3P evolution algorithm.

```
procedure EVOLVE()
    I ← INITPOPULATION(n_pop)
    𝓘 ← PARTITIONPOPULATION(I, EQ)
    n ← 0
    while n < n_gen n_pop do
        o ← GETOPERATOR(p_op,cross)
        G_p ← ∅
        while |G_p| < ARITY(o) do
            I' ← ∅
            for I ∈ 𝓘 do
                I' ← I' ∪ PARREP(I)
            end for
            (g_p, p_p, f_p) ← SELPAR(I')
            G_p ← G_p ∪ {g_p}
        end while
        G_c ← APPLY(o, G_p)
        for all g_c ∈ G_c do
            p_c ← MAP(g_c)
            f_c ← FITNESS(p_c)
            if ∃I_c ∈ 𝓘, ∃i ∈ I_c : EQ((g_c, p_c, f_c), i) then
                I_c ← I_c ∪ {(g_c, p_c, f_c)}
                while |I_c| > n_part do
                    I_c ← I_c \ REMOVALREP(I_c)
                end while
            else
                𝓘 ← 𝓘 ∪ {{(g_c, p_c, f_c)}}
            end if
        end for
        while |𝓘| > n_pop do
            I' ← ∅
            for I ∈ 𝓘 do
                I' ← I' ∪ REMOVALREP(I)
            end for
            i ← SELREMOVAL(I')
            𝓘 ← 𝓘 \ {I ∈ 𝓘 : i ∈ I}
        end while
        n ← n + |G_c|
    end while
end procedure
```

individuals equivalent if they have the same genotype, or phenotype, or fitness. We remark that in the three cases all the individuals in the same niche have the same fitness, as long as the fitness computation and mapping functions are deterministic.

Concerning the parent representative criterion PARREP(), we explored 5 options:

Uniform (U) one random individual is selected with uniform probability;

Youngest (A^-) the youngest individual is selected;

Oldest (A^+) the oldest individual is selected;

Shortest (L^-) the individual with shortest phenotype is selected;

Longest (L^+) the individual with longest phenotype is selected.

A^- and A^+ consider the age as the difference between the current generation index $\frac{n}{n_{\text{pop}}}$ and the generation index in which the individual was born. L^- and L^+ options consider the length of the phenotype, i.e., of the string of the language $\mathcal{L}(\mathcal{G})$. All the options with the exception of U may result in a tie, when applied: in that case, we choose a random individual (with uniform probability) among the ties.

The 3 different options for EQ() can be combined arbitrarily with the 5 different options for PARREP(), however some combinations result in the same outcome. In particular, with EQ() operating on the genotype, all the 5 options are equivalent, since the genetic operators will be applied on the same genotype, regardless of the chosen option: we hence considered only the U option, resulting in the combination denoted PPG3P-G-U in the experimental assessment. With EQ() operating on the phenotype, L^- nor L^+ always result in a tie, because all the individuals have the same length: we hence considered only U, A^-, and A^+, resulting in the combinations denoted PPG3P-P-U, PPG3P-P-A^-, and PPG3P-P-A^+. With EQ() operating on the fitness, we considered all the 5 options resulting in the combinations denoted PPG3P-F-U, PPG3P-F-A^-, PPG3P-F-A^+, PPG3P-P-L^-, and PPG3P-P-L^+.

Finally, concerning the representative removal criterion REMOVALREP(), we found from preliminary experimentation that the impact of this component on the evolution is negligible with respect to the other two components. We decided to define REMOVALREP() so as to select the oldest individual in the niche.

4.2. Deterministic Crowding G3P

Deterministic crowding is a variant of the original steady-state evolutionary algorithm that aims at promoting diversity by making the offspring compete with the parents for survival [17, 15]: a child replaces a parent only if fitter. The general working scheme of deterministic crowding does not specify how to decide which of

the offspring replaces which parent, but assumes that a similarity measure is used [17]. In this section we describe our adaptation of this strategy to the case of indirect representation, that we call *Deterministic Crowding* G3P (DCG3P). We make each child compete with the closest parent, the distance being measured at the level of genotype, or phenotype, or fitness. DCG3P thus requires a predefined *distance definition* between pairs of individuals (DIST() function in the description below).

The evolution algorithm is as follows (Algorithm 3). After the initialization of the population composed of n_{pop} individuals, the following steps are iterated.

1. The genetic operator is selected randomly between crossover and mutation (GETOPERATOR() function) with probability $p_{\text{op,cross}}$ and $1-p_{\text{op,cross}}$, respectively.
2. The proper number of parents with respect to the selected genetic operator are selected from the population, according to the parent selection function SELPAR().
3. The genetic operator is applied to the parent genotypes, resulting in one or more new children genotypes: the children genotypes are then mapped to phenotypes according to the genotype-phenotype mapping function MAP() and their fitness is computed using the FITNESS() function.
4. Each child is compared to the closest parent, the distance being measured by means of a function DIST(). If the child is fitter than the closest parent, than the child is added to the population, the parent is removed from the population and will not be considered when choosing the closest parent for next children. Otherwise, if the child is not fitter than the closest parent, then the child is not added to the population.

The iterative procedure is repeated until $n_{\text{gen}}n_{\text{pop}}$ individuals have been generated, much like SSG3P and PPG3P.

Concerning the distance function DIST() for determining the closest parent, we explored 3 options:

Genotype (G) Hamming distance among the genotypes of the two individuals (denoted DCG3P-G in the experimental assessment);

Phenotype (P) edit distance among the phenotypes (i.e, strings of the language $\mathcal{L}(\mathcal{G})$, denoted DCG3P-P);

Fitness (F) euclidean distance among the fitness values (i.e., $|f_c - f_p|$, denoted DCG3P-F).

Algorithm 3 DCG3P evolution algorithm.

```
procedure EVOLVE()
    I ← INITPOPULATION(n_pop)
    n ← 0
    while n < n_gen n_pop do
        o ← GETOPERATOR(p_op,cross)
        G_p ← ∅
        while |G_p| < ARITY(o) do
            (g_p, p_p, f_p) ← SELPAR(I)
            G_p ← G_p ∪ {g_p}
        end while
        G_c ← APPLY(o, G_p)
        for all g_c ∈ G_c do
            p_c ← MAP(g_c)
            f_c ← FITNESS(p_c)
            (g_p, p_p, f_p) ← CLOSEST((g_c, p_c, f_c), G_p, DIST)
            if f_c > f_p then
                I ← I \ {(g_p, p_p, f_p)}
                G_p ← G_p \ {(g_p, p_p, f_p)}
                I ← I ∪ {(g_c, p_c, f_c)}
            end if
        end for
        n ← n + |G_c|
    end while
end procedure
```

5. Experimental evaluation

We performed experiments in two phases: first, in order to experimentally verify to which degree the lack of diversity is an issue in existing G3P approaches, we performed a set of experiments with SSG3P and measured the diversity at the level of genotype, phenotype, and fitness; then, we compared the effectiveness of the two considered diversity promotion strategies (PPG3P and DCG3P) against the one of SSG3P, considered as a baseline.

5.1. Benchmark problems

We experimented on a set of 8 benchmark problems including Boolean, synthetic, and symbolic regression problems. We assembled this set by considering the guidelines for the evaluation of Genetic Programming approaches proposed in [49, 50] and the peculiarity of G3P.

- MOPM-3: Multiple outputs parallel 3-bit multiplier. The fitness is given by the number of errors among all the input cases.
- Parity-5 and Parity-8: 5- and 8-bit parity. The fitness is given by the number of errors among all the input cases.
- KLandscapes-7: a tunable (here $k = 7$), GP-specific benchmark [51]. We here adapted the fitness function to be consistent with the other problems (the lower, the better) by using $f(t) = 1-f_0(t)$, where $f_0(t)$ is the original fitness function described in the cited paper.
- Text [6]: generation of the target string Hello world!, where the fitness is the edit distance to the target string.
- Keijzer6 [52]: symbolic regression of the function $f(x) = \sum_{i=1}^{x} \frac{1}{i}$ on 50 points evenly spaced in $[1, 50]$ (50 points evenly spaced in $[1, 120]$ for validation)—note that validation points do not include learning points other than $x = 1$, even if they, in part, belong to the same interval $[1, 50]$.

- Nguyen7 [53]: symbolic regression of the function $f(x) = \log(x+1) + \log(x^2+1)$ on 20 points uniformly sampled in $[0, 2]$.
- Pagie1 [54]: symbolic regression of the function $f(x, y) = \frac{1}{1+x^{-4}} + \frac{1}{1+y^{-4}}$ on 125 points resulting from 25 values evenly spaced in $[-5, 5]$ for both x and y (10 000 points resulting from 100 values evenly spaced in the same interval for validation).

For the three symbolic regression problems, the fitness is given by the sum of the absolute errors between target and obtained values. The CFGs for all the benchmark problems are shown in Figure 2.

5.2. Diversity in SSG3P

We first performed a set of experiments by using SSG3P to provide the proper context and measure the diversity at the three levels, namely genotype, phenotype, and fitness. We quantify *diversity of the population* as the ratio between the number of different instances of the corresponding level and the population size—e.g., the phenotype diversity is the ratio between the number of different phenotypes in the population and the population size. Other means of measuring the population diversity could have been used (e.g., [55]): we chose this metric for its simplicity and wide adoption.

We performed 40 evolutionary runs for each problem and each representation variant (CFGGP, GE, SGE, and WHGE), with the evolutionary parameters shown in Table 1. Concerning the representation related parameters, we set $d = 12$ for CFGGP, $n = 8$, $n_w = 5$, $|g| = 1024$ for GE, $d_{\max} = 6$ for SGE, and $n_d = 3$, $|g| = 1024$ for WHGE: we chose these values by considering common practices and/or suggestions contained in the corresponding original publications. All the diversity measurements reported below are averaged across the 40 runs.

Table 1: Evolutionary parameters of SSG3P.

Population	$n_{\text{pop}} = 500$
Pop. init.	Random (GE, WHGE, SGE) Ramped half-and-half (CFGGGP)
Generations	50
Crossover op.	two-points same (GE, WHGE) SGE crossover (SGE) CFGGP crossover (CFGGP)
Gen. op. prob.	$p_{\text{op,cross}} = 0.8$, $1 - p_{\text{op,cross}} = 0.2$
Mutation op.	bit flip w. $p_{\text{mut}} = 0.01$ (GE, WHGE) SGE mutation w. $p_{\text{mut}} = 0.01$ (SGE) CFGGP mutation (CFGGP)
Parent sel.	tournament with size 5
Unsurv. sel.	worst fitness

Table 2 provides the genotype, phenotype, and fitness diversity in the initial and final population, i.e., after the last generation. Diversity at the genotype level is not available for CFGGP, because in this case the representation is direct.

The table shows that the initial genotype diversity is very high (equal to 1 in all cases). The fact that the phenotype diversity is different than the genotype diversity and, most importantly, significantly smaller is a clear indication of the degeneracy of the representation, i.e., multiple different genotypes are mapped to the same phenotype. Since the measure is obtained at the beginning of the evolution, it cannot be attributed to the effect of selective pressure, and the difference in diversity at the levels of genotype and phenotype may only be explained in terms of the genotype-phenotype mapping function. Table 2 also shows that the phenotype diversity greatly varies among problems and representations. The values for SGE and WHGE are similar and related to the complexity of the grammar of the problem: as suggested by common sense, the more complex the grammar, the greater the diversity. A similar trend is observed also for CFGGP, despite the fact that this representation is direct. Interestingly, the phenotype diversity for GE is much lower than the one observed for the other representations in the Boolean problems. This fact may be explained by the fact that, because of the structure of the corresponding CFGs, the event of mapping to an invalid phenotype is much more frequent: as a consequence, the number of different phenotypes is particularly low. This finding is consistent with the observations of [6].

Concerning the fitness diversity, it can be seen that it is, in all cases, much lower than the phenotype diversity. Interestingly, WHGE difference between phenotype and fitness diversity is lower than the one of SGE in many cases—e.g., in the Text problem, WHGE scores 0.91 in phenotype diversity and 0.14 in fitness diversity, whereas SGE scores 0.93 and 0.01, respectively. This could suggest that the differences among the phenotypes obtained with WHGE allow for a wider range of fitness values than SGE, that is, the two representations exhibit a different kind of phenotype diversity which cannot be captured by simply considering the rate of unique phenotypes. We argue that more sophisticated techniques for analysis, possibly based on visualization as the recently proposed DU map [16] or on different measures of diversity as in [55], might make more apparent those fine differences.

Table 2 also shows the final diversity, i.e., the diversity measured at the end of the evolution. With SGE the phenotype diversity remarkably decreases for all problems, a result that is coherent with its original aim of increasing locality—with high locality, there is a high correlation of the decrease of genotypic diversity and phenotypic diversity. As a consequence, also the fitness diversity is very low: in most problems, there is (on average across the 40 runs) only one unique phenotype at the end of the evolution. For the other representations, there is still a clear

MOPM-3

⟨o⟩ ::= ⟨e⟩ ⟨e⟩ ⟨e⟩ ⟨e⟩ ⟨e⟩ ⟨e⟩
⟨e⟩ ::= .or ⟨e⟩ ⟨e⟩ | .xor ⟨e⟩ ⟨e⟩ | .and ⟨e⟩ ⟨e⟩ |
.and1not ⟨e⟩ ⟨e⟩ | ⟨v⟩
⟨v⟩ ::= v1.1 | v1.2 | v1.3 | v2.1 | v2.2 | v2.3

Parity-5 and Parity-8

gft⟨e⟩ ::= .or ⟨e⟩ ⟨e⟩ | .and ⟨e⟩ ⟨e⟩ | .not ⟨e⟩ | ⟨v⟩
⟨v⟩ ::= v1 | v2 | v3 | v4 | v5 (and | v6 | v7 | v8)

Keijzer6

⟨expr⟩ ::= ⟨op⟩ ⟨expr⟩ ⟨expr⟩ | pre-op ⟨expr⟩ | ⟨var⟩
⟨op⟩ ::= + | *
⟨pre-op⟩ ::= uminus | 1/ | sqrt
⟨var⟩ ::= x

Nguyen7

⟨expr⟩ ::= ⟨op⟩ ⟨expr⟩ ⟨expr⟩ | pre-op ⟨expr⟩ | ⟨var⟩
⟨op⟩ ::= + | - | p/ | *
⟨pre-op⟩ ::= sin | cos | exp | plog
⟨var⟩ ::= x | 1.0

Pagie1

⟨expr⟩ ::= ⟨op⟩ ⟨expr⟩ ⟨expr⟩ | pre-op ⟨expr⟩ | ⟨var⟩
⟨op⟩ ::= + | - | p/ | *
⟨pre-op⟩ ::= sin | cos | exp | plog
⟨var⟩ ::= x | y | 1.0

KLandscapes-7

⟨N⟩ ::= ⟨n⟩ ⟨N⟩ ⟨N⟩ | ⟨t⟩
⟨n⟩ ::= n0 | n1
⟨t⟩ ::= t0 | t1 | t2 | t3

Text

⟨text⟩ ::= ⟨sentence⟩ ⟨text⟩ | ⟨sentence⟩
⟨sentence⟩ ::= ⟨Word⟩ ␣ ⟨sentence⟩ | ⟨word⟩ ␣ ⟨sentence⟩ |
⟨word⟩ ⟨punct⟩
⟨word⟩ ::= ⟨letter⟩ ⟨word⟩ | ⟨letter⟩
⟨Word⟩ ::= ⟨Letter⟩ ⟨word⟩
⟨letter⟩ ::= ⟨vowel⟩ | ⟨consonant⟩
⟨vowel⟩ ::= a | e | i | o | u
⟨consonant⟩ ::= b | c | d | ... | z
⟨Letter⟩ ::= ⟨Vowel⟩ | ⟨Consonant⟩
⟨Vowel⟩ ::= A | E | I | O | U
⟨Consonant⟩ ::= B | C | D | ... | Z
⟨punct⟩ ::= ! | ? | .

Figure 2: The grammars of the benchmark problems: in the symbolic regression problems, p/ and plog are the protected versions of the division and the logarithm, respectively.

Table 2: Initial Diversity (ID) and Final Diversity (FD) with SSG3P, mean value across the runs.

		Boolean			Symbolic regression			Synthetic	
		MOPM-3	Parity-5	Parity-8	Keijzer6	Nguyen7	Pagie1	KLand.-7	Text
	Repr.	ID → FD	ID → FD	ID → FD	ID → FD	ID → FD	ID → FD	ID → FD	ID → FD
Geno.	CFGGP	1.00→0.45	0.49→0.66	0.52→0.59	0.43→0.00	0.49→0.02	0.52→0.06	0.4 →0.08	0.96→0.01
	GE	1.00→0.99	1.00→0.98	1.00→0.99	1.00→0.91	1.00→0.91	1.00→0.94	1.00→0.95	1.00→0.98
	SGE	1.00→0.05	1.00→0.02	1.00→0.02	1.00→0.07	1.00→0.12	1.00→0.13	1.00→0.02	1.00→0.14
	WHGE	1.00→0.88	1.00→0.83	1.00→0.79	1.00→0.89	1.00→0.81	1.00→0.78	1.00→0.92	1.00→0.92
Pheno.	CFGGP	1.00→0.45	0.49→0.66	0.52→0.59	0.43→0.00	0.49→0.02	0.52→0.06	0.4 →0.08	0.96→0.01
	GE	0.00→0.00	0.06→0.05	0.07→0.07	0.36→0.07	0.43→0.04	0.46→0.05	0.35→0.1	0.95→0.01
	SGE	1.00→0.02	0.81→0.00	0.83→0.00	0.5 →0.00	0.54→0.00	0.56→0.00	0.43→0.00	0.92→0.01
	WHGE	1.00→0.63	0.79→0.37	0.81→0.25	0.62→0.1	0.66→0.04	0.64→0.23	0.44→0.64	0.91→0.02
Fitness	CFGGP	0.02→0.00	0.02→0.00	0.03→0.00	0.24→0.00	0.39→0.00	0.41→0.00	0.21→0.00	0.07→0.00
	GE	0.00→0.00	0.00→0.00	0.00→0.00	0.20→0.05	0.34→0.03	0.36→0.01	0.18→0.1	0.11→0.00
	SGE	0.03→0.00	0.02→0.00	0.01→0.00	0.32→0.00	0.42→0.00	0.44→0.00	0.18→0.00	0.01→0.00
	WHGE	0.03→0.00	0.02→0.00	0.03→0.01	0.18→0.09	0.56→0.02	0.52→0.11	0.24→0.16	0.14→0.00

decrease in diversity at all levels: WHGE exhibits the largest phenotype and fitness diversities in the majority of the problems.

Figure 3 illustrates the relation between phenotype diversity and best fitness during the evolution. There is one plot for each problem; each plot contains four sets of points (in four different colors), one set for each representation and one point for each generation—earlier generations corresponding to smaller fitness values. The actual values for the final best fitness are reported in the next section (Table 4).

This figure makes the differences among representations apparent. CFGGP and WHGE tend to exhibit a gradual decrease of phenotype diversity and, at the same time, an improvement of the fitness. GE and SGE have instead a more varied behavior with several problems in which they tend to exhibit a fast reduction in the diversity in the first phase of the evolution. The figure also shows that GE struggles to improve the best fitness in many of the problems, whereas SGE still manages to obtain good final best fitness (with respect to the best value for each problem) even in spite of the fast initial decrease in phenotype diversity.

Finally, Figure 4 shows how the genotype, phenotype, and fitness diversity varies during the evolution, each curve corresponding to the value of diversity averaged across all problems (as well as across all runs). The figure essentially confirms the findings of Table 2 and further highlights the fact that the decrease in diversity in SGE is, at all levels, faster than in the other representations. On the other hand, Figure 4 shows that WHGE appears to be intrinsically more prone to the preservation of diversity. We explain this finding by the fact that this mapping variant exhibits low degeneracy [56] and redundancy [16] (i.e., it tends to exploit the full genotype during the mapping): this makes the chance rarer to obtain an identical individual upon the application of genetic operators.

In summary, it is fair to conclude that the results of these first experimental phase suggest that there is an issue concerning the lack of diversity in SSG3P. Although the lack of diversity is not, alone, an evidence of a premature convergence to a local optimum, we think that these results suggest that there is an opportunity for improving the EA effectiveness by employing a diversity promotion strategy, i.e., they lay the ground for PPG3P and DCG3P.

5.3. Assessment of diversity promotion strategies

In this section we discuss the experimental assessment of the diversity promotion strategies PPG3P and DCG3P. We used the same procedure as in the previous section, i.e., we measured diversity at the genotype, phenotype and fitness level in 40 independent runs and provide the average values. We configured PPG3P with $n_{\text{part}} = 20$, while DCG3P has no parameters.

We considered all the 9 PPG3P variants and all the 3 DCG3P variants described in Section 4.1 and in Section 4.2, respectively. PPG3P variants differ in whether

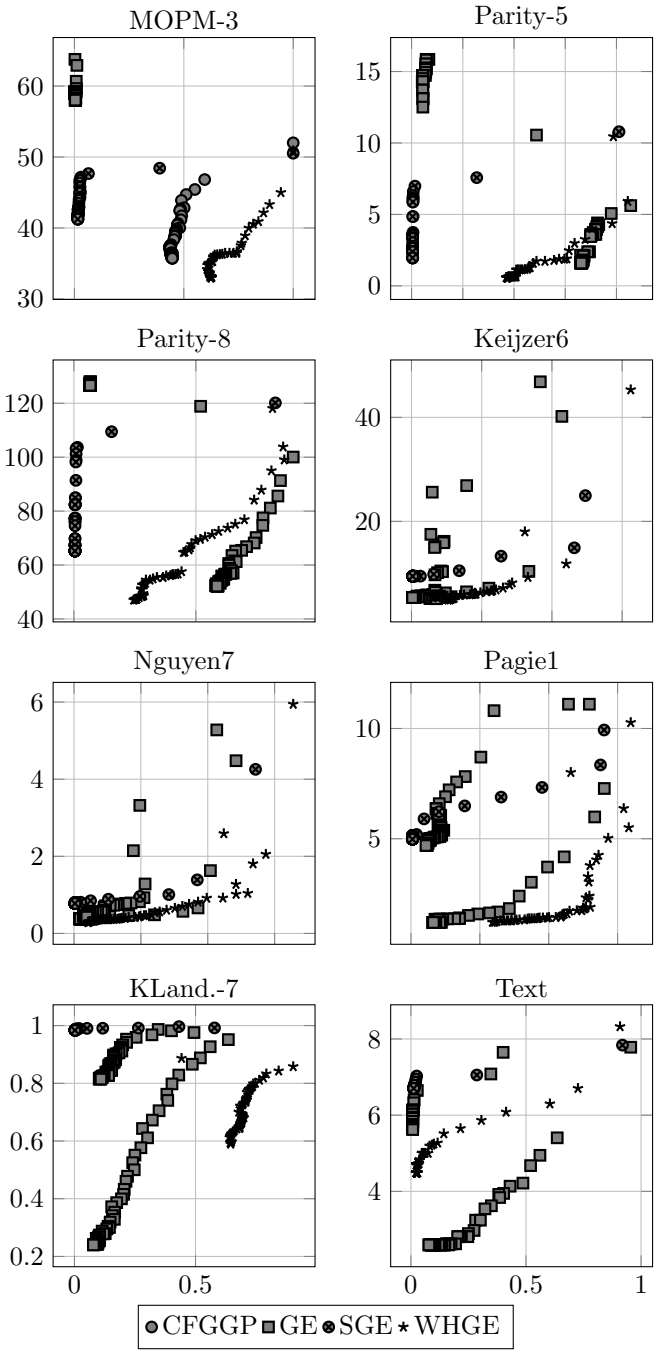

Figure 3: Best fitness (on y-axis) vs. phenotype diversity (on x-axis) in SSG3P, one mark for each generation: the mark coordinates are the average values of the two respective indexes across the runs at the corresponding generation.

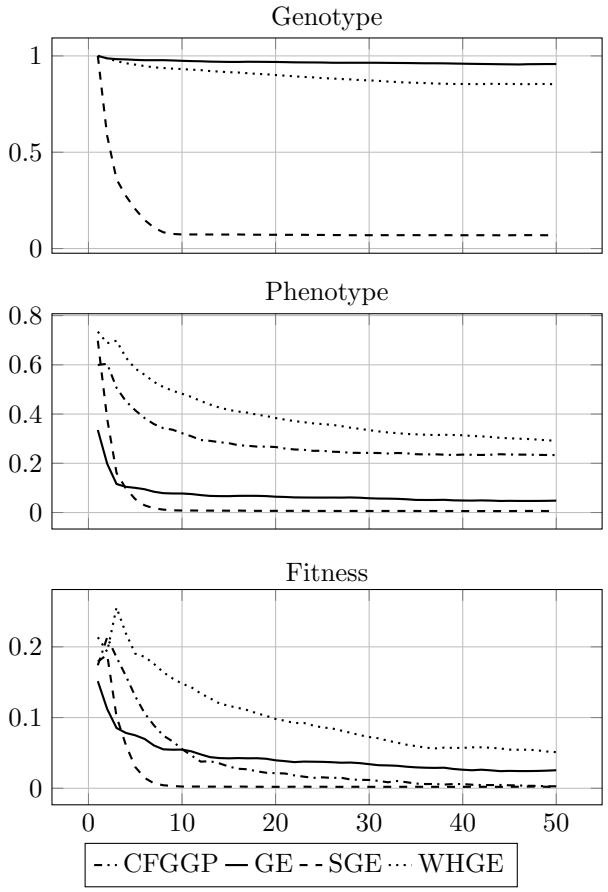

Figure 4: Average diversity in SSG3P during the evolution: values are averaged across runs and problems.

the equivalence relation for partitioning the population in niches is defined at the fitness, phenotype or genotype level (F, P, G, respectively), and in the criterion for selecting the parent within a niche (U, A^-, A^+, L^-, L^+). DCG3P variants differ in the distance definition used for determining the closest parent, which may be either Euclidean distance between fitness values (F), or Edit distance between phenotypes (P), or Hamming distance between genotypes (G).

Table 3 provides the final diversity for each of the proposed diversity promotion approaches, measured at the genotype level (upper portion), phenotype level (middle portion), and fitness level (lower portion). Diversity values are averaged across all problems, for ease of presentation. The corresponding value in the absence of any diversity promotion approach is provided in the topmost row of each portion (SSG3P). Values in bold indicate the highest diversity value for each column and each diversity definition: for example, 0.996 in the SGE column, first group of rows, is the highest diversity value measured at the genotype level for SGE.

It can be seen that all the proposed diversity promotion approaches are indeed highly beneficial, as they all tend to improve diversity significantly in all cases, that is, irrespective of how diversity is quantified and irrespective of the evolutionary strategy. There are very few exceptions to this summary: PPG3P does not improve diversity with CFGGP when measuring diversity at the phenotype level; PPG3P-G-U does not improve diversity with WHGE when measuring diversity at the genotype level. It can also be seen that the improvement in diversity tends to be most significant at the phenotype level. Concerning the diversity improvement across the four mapping variants, the improvement is particularly significant with SGE at all the three levels—as shown in Section 5.2, SGE tends to exhibit the lowest diversity values.

Table 4 provides the final best fitness separately for each problem. This table consists of 4 portions, one for each of the mapping variants considered; the topmost row of each portion (SSG3P) corresponds to not using any of the proposed diversity promotion approaches (i.e., SSG3P rows corresponds to the experiments of the previous Section).

It can be seen that all the proposed diversity promotion approaches tend to be beneficial also in terms of final fitness values. The results in this case are more nuanced, however. The final best fitness worsens with the KLand.-7 problem, with the MOPM-3 problem (only with CFGGP), and with the Pagie1 problem (only with WHGE). For all the other problems and mapping variants, one of the diversity promotion strategies delivers a final best fitness value better than the one obtained without diversity promotion.

A summary of the results concerning the final best fitness is provided by Table 5, which shows the average percentile rank of each combination of a strategy and a mapping variant. More in detail, (i) we considered separately the 8 problems and, for each one, we obtained the percentile rank of each evolutionary run among all

Table 3: Final diversity at the three levels (genotype, phenotype, fitness) for each diversity promotion strategy: values are averaged across runs and problems.

	Strategy	CFGGP	GE	SGE	WHGE
Genotype	SSG3P	0.257	0.959	0.068	0.85
	PPG3P-G-U	0.201	0.756	0.075	0.79
	PPG3P-P-U	0.222	0.966	0.324	0.884
	PPG3P-P-A$^-$	0.194	0.965	0.336	0.875
	PPG3P-P-A$^+$	0.208	0.968	0.234	0.869
	PPG3P-F-U	**0.583**	0.976	0.535	0.957
	PPG3P-F-A$^-$	0.541	0.969	0.404	0.952
	PPG3P-F-A$^+$	0.517	0.973	0.427	0.94
	PPG3P-F-L$^-$	0.35	0.972	0.369	0.933
	PPG3P-F-L$^+$	0.48	0.944	0.336	0.914
	DCG3P-G	0.572	**1**	**0.996**	**1**
	DCG3P-P	0.57	**1**	0.925	**1**
	DCG3P-F	0.573	**1**	0.859	**1**
Phenotype	SSG3P	0.257	0.048	0.006	0.293
	PPG3P-G-U	0.201	0.057	0.008	0.295
	PPG3P-P-U	0.222	0.186	0.098	0.42
	PPG3P-P-A$^-$	0.194	0.175	0.1	0.422
	PPG3P-P-A$^+$	0.208	0.176	0.095	0.412
	PPG3P-F-U	**0.583**	0.285	0.326	**0.712**
	PPG3P-F-A$^-$	0.541	0.259	0.227	0.703
	PPG3P-F-A$^+$	0.517	0.307	0.318	0.649
	PPG3P-F-L$^-$	0.35	0.243	0.187	0.428
	PPG3P-F-L$^+$	0.48	**0.426**	0.239	0.677
	DCG3P-G	0.572	0.289	**0.622**	0.664
	DCG3P-P	0.57	0.303	0.597	0.664
	DCG3P-F	0.573	0.306	0.519	0.646
Fitness	SSG3P	0.003	0.023	0.002	0.047
	PPG3P-G-U	0.048	0.035	0.001	0.039
	PPG3P-P-U	0.049	0.113	0.019	0.083
	PPG3P-P-A$^-$	0.051	0.107	0.02	0.092
	PPG3P-P-A$^+$	0.05	0.116	0.022	0.093
	PPG3P-F-U	0.102	0.142	0.074	**0.158**
	PPG3P-F-A$^-$	0.101	**0.154**	0.074	0.158
	PPG3P-F-A$^+$	0.1	0.142	0.073	0.152
	PPG3P-F-L$^-$	0.102	0.139	0.072	0.145
	PPG3P-F-L$^+$	0.1	0.152	0.073	0.156
	DCG3P-G	0.144	0.097	**0.091**	0.109
	DCG3P-P	0.139	0.11	0.091	0.1
	DCG3P-F	**0.145**	0.117	0.071	0.087

the $40 \times 4 \times 13$ runs with that problem, (ii) we averaged the ranks for each strategy-mapping combination across different runs and problems. For example, 32 for SSG3P-CFGGP means that that combination delivers, on average across runs and problems, a final best fitness value which places 32-nd on 100.

Table 5 essentially confirms the findings suggested by the previous tables. First, some form of diversity promotion is beneficial to the effectiveness of the evolutionary search (i.e., results in a better final best fitness) in every case, regardless of the mapping variant. Second, the improvement is in general negligible with WHGE; only one strategy (PPG3P-P-A$^-$) actually leads to an improvement with this mapping variant: we speculate that this is related to the fact that WHGE has intrinsically larger diversity (see Figure 4) and hence less room for improvements. Third, promoting the diversity at the phenotype level seems to be more effective than at the level of genotype and fitness, in particular with PPG3P: by looking at the raw results, we observed that enforcing diverse fitness values often turned out to be a too aggressive strategy, resulting in poorly performing individuals being (relatively) over-represented in the population at the expense of good individuals. Considering the different options for the parent representative criterion (ParRep()) in PPG3P-P, no sharp conclusions can be drawn: U works better with CFGGP and SGE (tie), A$^-$ with SGE and WHGE, A$^+$ with GE; overall, however, differences are negligible. Finally, the DCG3P strategy appears to work poorly when coupled to GE and WHGE: we think that this finding might be explained by the fact that these two mappings exhibit a lower locality than the others [56] and that DCG3P is based on a distance, differently than PPG3P.

Table 5 concisely suggests that the more promising strategies are PPG3P-P-U, PPG3P-P-A$^-$, PPG3P-P-A$^+$ and DCG3P-F, though they perform better with different mapping variants. For these strategies and for the baseline SSG3P, we analyzed the statistical significance of the results. We performed a non-parametric Friedman test in order to assess the existence of statistically significant differences between those selected strategies: the p-value for each mapping variant is definitely lower than a confidence $\alpha = 0.05$ (0.0014, 3×10^{-8}, 2×10^{-16}, 2×10^{-9}, respectively for CFGGP, GE, SGE, and WHGE). Therefore, we conducted a suite of post-hoc tests for multiple comparison for discerning which pairs of strategies are significant different. Table 6 reports the adjusted p-values computed through Holm's procedure: the values highlighted in bold indicate whether the comparison is statistically significant with a confidence $\alpha = 0.05$. These results corroborate most of the considerations derived from Table 5. In particular, the comparison with DCG3P-F or with the baseline SSG3P shows significant differences. Concerning the mapping variants, we can draw firm conclusions regarding GE and SGE, whereas for WHGE we have a statistically strong evidence only with respect to the DCG3P-F strategy.

Table 4: Final best fitness: mean and standard deviations across runs.

		Boolean			Symbolic regression			Synthetic	
	Strategy	MOPM-3	Parity-5	Parity-8	Keijzer6	Nguyen7	Pagie1	KLand.-7	Text
CFGGP	SSG3P	**36.1**±4.4	1.5±1.2	51.8±18.5	5.41±1.17	0.38±0.14	1.28±0.82	**0.25**±0.17	2.5±1.1
	PPG3P-P-U	37.6±3.9	1.2±1.6	50 ±16.6	3.99±0.76	0.14±0.1	0.76±0.5	0.43±0.07	1.3±0.9
	PPG3P-P-A$^-$	38.7±2.8	1.3±1.3	42.6±17.8	4.23±0.72	**0.12**±0.08	0.75±0.87	0.46±0.06	1.6±1.1
	PPG3P-P-A$^+$	38.6±3.6	0.9±0.9	43.8±24.7	4.45±0.86	0.12±0.07	0.78±0.46	0.43±0.07	1.7±0.9
	PPG3P-F-U	40.6±4.6	1.2±1.4	52.9±22.4	4.21±1.09	0.15±0.08	1.62±1.37	0.5 ±0.04	3.3±1.3
	PPG3P-F-A$^-$	39.8±3.1	1.4±1.6	58.6±17.3	4.21±1.99	0.13±0.09	1.15±0.61	0.5 ±0.06	3.4±1.1
	PPG3P-F-A$^+$	43.5±2.5	1.6±2.2	69.1±17.3	4.37±0.83	0.2 ±0.08	1.08±0.44	0.46±0.05	4.4±1
	PPG3P-F-L$^-$	41.9±6.3	1.8±2.9	**27.6**±30	4.74±1.26	0.15±0.09	0.79±0.38	0.49±0.08	6.1±1.1
	PPG3P-F-L$^+$	38.8±2.9	**0.2**±0.7	49.5±28.4	4.69±1.07	0.18±0.1	0.82±0.67	0.44±0.06	3 ±1.1
	DCG3P-P	36.8±3.6	0.4±0.8	42.4±16.3	5.06±0.77	0.28±0.07	1.25±0.27	0.52±0.02	**0.5**±0.6
	DCG3P-F	37.4±2.2	0.3±0.8	44.1±26.5	**3.57**±1.18	0.19±0.08	**0.67**±0.26	0.48±0.05	1.5±0.8
GE	SSG3P	58.1±7.9	12.7±5.6	125.7±12.1	5.15±2.23	0.4 ±0.24	4.66±2.61	**0.82**±0.05	5.6±0.7
	PPG3P-G-U	54 ±7.5	12.3±6.4	128 ± 0	5.85±1.58	0.39±0.31	3.1 ±1.96	0.83±0.05	5 ±1.4
	PPG3P-P-U	**51.5**±5.4	1.7±2.5	116 ±22.3	5.87±2.02	0.43±0.18	3.45±1.87	0.88±0.02	2.9±0.9
	PPG3P-P-A$^-$	55.8±6.4	**0.3**±1.5	121.2±18.3	5.66±1.2	**0.28**±0.15	3.8 ±1.14	0.88±0.02	3.3±1.1
	PPG3P-P-A$^+$	60.5±7.6	0.6±2	114.8±32.9	5.43±1.85	0.34±0.2	3.27±1.93	0.87±0.03	**2.8**±1
	PPG3P-F-U	54.1±4.5	12.7±6	128 ± 0	5.83±1.55	0.55±0.34	**2.59**±1.54	0.88±0.03	5.8±0.8
	PPG3P-F-A$^-$	63.6±4.8	15.2±3.1	125.6± 5.8	6.12±2.19	0.37±0.19	3.08±1.08	0.88±0.03	6.4±0.8
	PPG3P-F-A$^+$	63.9±3.5	16 ±0	128 ± 0	6.06±1.66	0.85±0.68	3.74±1.27	0.88±0.02	6.3±0.7
	PPG3P-F-L$^-$	63.6±4.2	12.3±6.8	128 ± 0	6.82±2.51	0.3 ±0.19	2.74±1.48	0.9 ±0.03	6.7±0.5
	PPG3P-F-L$^+$	59 ±8	1.7±2.4	**98.5**±19.5	**4.66**±0.8	0.31±0.14	3.31±1.63	0.9 ±0.03	4.9±0.8
	DCG3P-G	60.8±7.1	11.1±7.2	128 ± 0	6.47±0.84	0.5 ±0.16	4.25±1.05	0.87±0.02	4.9±0.6
	DCG3P-P	52.6±7.1	10.4±6.7	128 ± 0	6.18±0.88	0.52±0.18	3.46±2.06	0.88±0.02	4.9±0.8
	DCG3P-F	54.5±5.3	14.6±3.8	128 ± 0	5.43±0.79	0.5 ±0.32	2.77±0.91	0.86±0.03	5.1±0.6
SGE	SSG3P	40.9±5.1	2.1±2.4	63.6±31.4	9.81±3.04	0.78±0.11	5.05±2.05	0.98±0	6.7±0.6
	PPG3P-G-U	42.8±3.8	**0** ±0	53.2±38.7	9.07±3.21	0.69±0.11	5.62±2.41	**0.98**±0	6.6±0.5
	PPG3P-P-U	**36.5**±5.5	**0** ±0	20.8±28.5	6.76±0	0.54±0.08	2.4 ±0.66	0.98±0	6.1±0.3
	PPG3P-P-A$^-$	38.7±4.3	**0** ±0	**18** ±30.7	6.8 ±0.24	0.52±0.16	2.81±0.5	0.98±0	**6** ±0.2
	PPG3P-P-A$^+$	39.1±3	0.1±0.5	23 ±34.1	**6.73**±0.21	0.57±0.06	**2.24**±0.65	0.98±0	6.2±0.4
	PPG3P-F-U	42.6±2.7	1.2±2.8	50.6±33.4	6.91±0.71	**0.44**±0.12	2.48±0.46	0.98±0	7.3±0.6
	PPG3P-F-A$^-$	46.4±1.7	2.7±3.6	56.4±31.5	6.99±0.77	0.56±0.08	2.89±0.93	0.98±0	7.3±0.5
	PPG3P-F-A$^+$	44.5±2.7	0.6±1.7	60.4±31.8	6.76±0	0.53±0.09	2.85±0.69	0.98±0	7.2±0.7
	PPG3P-F-L$^-$	46.6±1.3	1.3±2.5	73.6±35	6.75±0.74	0.52±0.12	2.36±0.48	0.98±0	7.7±0.5
	PPG3P-F-L$^+$	43.5±2.1	2.4±2.2	46.7±21.5	6.82±0.7	0.54±0.09	2.39±0.72	0.98±0	7.1±0.6
	DCG3P-G	38.8±3.3	**0** ±0	26 ±30.7	6.79±0.21	0.65±0.04	2.65±0.5	0.99±0	6 ±0
	DCG3P-P	41.6±2.2	**0** ±0	51.1±35.7	6.76±0	0.57±0.13	2.94±0.55	0.98±0	6 ±0
	DCG3P-F	43.2±2.4	0.3±1	33.4±31.4	6.87±0.67	0.66±0.04	3.39±0.89	0.98±0	6 ±0
WHGE	SSG3P	33.1±4.6	0.6±0.6	47.6±29	4.85±0.55	0.29±0.11	**1.22**±0.39	**0.59**±0.02	4.5±0.6
	PPG3P-G-U	33 ±2.6	0.7±0.6	49.8±26.4	**3.88**±1.49	0.35±0.12	1.38±0.58	0.6 ±0.03	4.5±0.9
	PPG3P-P-U	33.8±3	0.7±0.8	48.8±21.6	5.03±0.83	0.29±0.08	1.76±0.65	0.62±0.03	**2.8**±1
	PPG3P-P-A$^-$	**30.6**±4.9	**0.3**±0.6	44.7±13.4	5.56±0.95	0.33±0.09	1.37±0.42	0.63±0.02	3.2±1
	PPG3P-P-A$^+$	35 ±2	1.1±1.4	42.4±17.8	5 ±1.07	0.29±0.09	1.41±0.64	0.63±0.03	3.1±1.1
	PPG3P-F-U	36.2±2.3	1.2±0.9	57 ±18.8	5.54±0.63	**0.28**±0.1	1.52±0.42	0.68±0.02	5.5±1.1
	PPG3P-F-A$^-$	35.6±2.7	1.1±1.4	57 ±22.8	5.52±0.35	0.34±0.08	1.82±0.7	0.72±0.03	5.3±1.1
	PPG3P-F-A$^+$	38 ±3.8	1.9±1.6	64.3±12.8	5.62±0.75	0.32±0.09	1.5 ±0.57	0.71±0.03	6.5±0.7
	PPG3P-F-L$^-$	38.6±4.3	1.5±1.7	66.1±17	5.24±0.66	0.3 ±0.09	1.56±0.67	0.71±0.04	6.7±0.7
	PPG3P-F-L$^+$	36.8±2.1	0.5±0.9	63.1±17.6	5.03±0.77	0.37±0.16	1.31±0.48	0.68±0.02	5.2±0.5
	DCG3P-G	36.5±2.5	0.4±0.8	53.3±17.6	5.56±0.83	0.54±0.12	2.24±0.69	0.71±0.02	4.5±0.7
	DCG3P-P	34.7±2.5	0.7±1.1	**40.9**±25.5	4.78±1.22	0.5 ±0.09	2.45±0.48	0.69±0.03	4.4±1
	DCG3P-F	35.3±1.6	0.7±1	45.6±19.6	5.04±0.93	0.35±0.12	1.9 ±0.72	0.68±0.03	4.2±0.7

Table 5: Average percentile rank of final best fitness (see text).

Strategy	CFGGP	GE	SGE	WHGE
SSG3P	32	65	69	30
PPG3P-G-U		62	61	31
PPG3P-P-U	20	55	**47**	30
PPG3P-P-A$^-$	23	52	**47**	**28**
PPG3P-P-A$^+$	22	**50**	49	30
PPG3P-F-U	29	66	56	40
PPG3P-F-A$^-$	28	70	65	40
PPG3P-F-A$^+$	34	74	60	46
PPG3P-F-L$^-$	29	66	64	43
PPG3P-F-L$^+$	23	54	62	36
DCG3P-G	27	70	51	41
DCG3P-P	24	66	54	38
DCG3P-F	**18**	66	56	34

Finally, we investigated the impact of the diversity promotion strategies on the generalization ability of the evolutionary search. To this end, we considered the error on the validation data for the two benchmark problems for which this was present (Keijzer6 and Pagie1). The results are shown in Table 7, which provides the mean and standard deviations across runs of this figure for each strategy-mapping combination.

The numbers of Table 7 show that promoting diversity is beneficial also in terms of generalization ability. As for final best fitness, the improvement is larger for SGE and CFGGP, though it is less evident which precise strategy actually delivers the larger improvement—the most convenient choice being dependent on both the problem and mapping variant.

6. Conclusions

Grammar-guided Genetic Programming (G3P) is a family of Evolutionary Algorithms for which the possibility of tackling different problems by simply changing the user-provided grammar resulted in large adoption and enduring popularity. The most widespread members of this family are based on an indirect representation: individuals are described at three levels in terms of their genotype, phenotype, and fitness. This representation stimulated criticism and attracted many studies: many of them found clues that lack of diversity in the population may be an intermediate effect of the representation properties.

In this paper, we experimentally analyzed in detail how population diversity, measured at the three levels mentioned above, changes during the evolution for 4 relevant G3P variants (CFGGP, GE, SGE, and WHGE) on different benchmark problems. We also considered two diversity promotion strategies which can operate at the three levels. We showed experimentally that some form of diversity promotion is always beneficial to the search effectiveness: best individuals at the last generation have a better fitness which also correspond to better generalization ability. We discussed which nuances of the 4 G3P variants may impact the different outcomes of the promotion of diversity.

We think that our results may make G3P approaches even more attractive to practitioners, because we show that promoting diversity can lead to substantial improvements in the fitness. The considered diversity promotion strategies could even be used as part of a standard toolbox for exploring different configurations in practical applications. From a broader perspective, we hope that our findings concerning the interaction between indirect representations and diversity may stimulate EA designers to take into account the promotion of diversity while building new EAs or adapting existing ones.

Acknowledgments

We would like to thank the anonymous reviewers for their insightful comments and constructive suggestions.

References

References

[1] J. R. Koza, Genetic programming: on the programming of computers by means of natural selection, Vol. 1, MIT press, 1992.

[2] R. I. Mckay, N. X. Hoai, P. A. Whigham, Y. Shan, M. O'neill, Grammar-based genetic programming: a survey, Genetic Programming and Evolvable Machines 11 (3-4) (2010) 365–396.

[3] P. A. Whigham, Inductive bias and genetic programming, in: First International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications, 1995, pp. 461–466. doi:10.1049/cp:19951092.

[4] A. Geyer-Schulz, Fuzzy rule-based expert systems and genetic machine learning, Vol. 3, Physica Verlag, 1997.

[5] C. Ryan, J. Collins, M. O'Neill, Grammatical evolution: Evolving programs for an arbitrary language, in: European Conference on Genetic Programming, Springer, 1998, pp. 83–96.

[6] E. Medvet, A comparative analysis of dynamic locality and redundancy in grammatical evolution, in: J. McDermott, M. Castelli, L. Sekanina, E. Haasdijk, P. García-Sánchez (Eds.), Genetic Programming, Springer International Publishing, Cham, 2017, pp. 326–342.

[7] R. Franz, Representations for Genetic and Evolutionary Algorithms, Springer Berlin Heidelberg, 2006. doi:10.1007/3-540-32444-5.
URL `https://doi.org/10.1007/3-540-32444-5`

[8] A. Brabazon, M. O'Neill, S. McGarraghy, Grammatical evolution, in: Natural Computing Algorithms, Springer Nature, 2015, pp. 357–373.

[9] T. Castle, C. G. Johnson, Positional effect of crossover and mutation in grammatical evolution, in: European Conference on Genetic Programming, Springer, 2010, pp. 26–37.

[10] N. Q. Uy, N. X. Hoai, M. O'Neill, R. I. McKay, D. N. Phong, On the roles of semantic locality of crossover in genetic programming, Information Sciences 235 (2013) 195–213.

[11] A. Thorhauer, F. Rothlauf, On the locality of standard search operators in grammatical evolution, in: International Conference on Parallel Problem Solving from Nature, Springer, 2014, pp. 465–475.

[12] M. O'Neill, A. Brabazon, M. Nicolau, S. Mc Garraghy, P. Keenan, πgrammatical evolution, in: Genetic and Evolutionary Computation Conference, Springer, 2004, pp. 617–629.

[13] N. Lourenço, F. B. Pereira, E. Costa, Unveiling the properties of structured grammatical evolution, Genetic Programming and Evolvable Machines (2016) 251–289.

Table 6: Adjusted p-values according to Holm's procedure for selected pairs of strategies. Highlighted values indicate statistically significance with $\alpha = 0.05$.

Pair of strategies		CFGGP	GE	SGE	WGHE
DCG3P-F	PPG3P-P-A$^-$	1	**0.0029**	**5e-6**	**6e-7**
DCG3P-F	PPG3P-P-A$^+$	1	**0.0007**	**0.0050**	**3e-5**
DCG3P-F	PPG3P-P-U	1	0	**3e-6**	**2e-6**
DCG3P-F	SSG3P	**0.0054**	0.8251	**8e-14**	**4e-8**
PPG3P-P-A$^-$	PPG3P-P-A$^+$	1	0.7745	0.2249	1
PPG3P-P-A$^-$	PPG3P-P-U	1	0.1813	0.9068	1
PPG3P-P-A$^-$	SSG3P	**0.0021**	**0.0175**	**0**	1
PPG3P-P-A$^+$	PPG3P-P-U	1	0.7745	0.2249	1
PPG3P-P-A$^+$	SSG3P	0.0561	**0.0003**	**0**	1
PPG3P-P-U	SSG3P	**0.0205**	**6e-6**	**0**	1

[14] E. Medvet, Hierarchical grammatical evolution, in: Proceedings of the Genetic and Evolutionary Computation Conference Companion, GECCO '17, ACM, New York, NY, USA, 2017, pp. 249–250. doi:10.1145/3067695.3075972.

[15] G. Squillero, A. P. Tonda, Divergence of character and premature convergence: A survey of methodologies for promoting diversity in evolutionary optimization, Information Sciences 329 (2016) 782–799. doi:10.1016/j.ins.2015.09.056.

[16] E. Medvet, M. Virgolin, M. Castelli, P. A. N. Bosman, I. Gonçalves, T. Tušar, Unveiling evolutionary algorithm representation with du maps, Genetic Programming and Evolvable Machines 19 (3) (2018) 351–389. doi:10.1007/s10710-018-9332-5.

[17] S. W. Mahfoud, Niching Methods for Genetic Algorithms, Ph.D. thesis, University of Illinois at Urbana-Champaign, Champaign, IL, USA, UMI Order No. GAX95-43663 (1995).

[18] E. Medvet, A. Bartoli, G. Squillero, An effective diversity promotion mechanism in grammatical evolution, in: Proceedings of the Genetic and Evolutionary Computation Conference Companion, ACM, 2017, pp. 247–248.

[19] J. H. Holland, Adaptation in Natural and Artificial Systems, University of Michigan Press, 1975.

[20] E. Mayr, Darwin's principle of divergence, Journal of the History of Biology 25 (3) (1992) 343–359.

[21] B. Sareni, L. Krahenbuhl, Fitness sharing and niching methods revisited, Evolutionary Computation, IEEE Transactions on 2 (3) (1998) 97–106.

[22] B. Kulis, Metric learning: A survey, Foundations and Trends in Machine Learning 5 (4) (2012) 287–364.

[23] A. Bellet, A. Habrard, M. Sebban, A survey on metric learning for feature vectors and structured data, arXiv preprint arXiv:1306.6709 (2013).

[24] E. Medvet, A. Bartoli, G. Davanzo, A. De Lorenzo, Automatic face annotation in news images by mining the web, in: Proceedings of the 2011 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology-Volume 01, IEEE Computer Society, 2011, pp. 47–54.

[25] P. M. Vitányi, F. J. Balbach, R. L. Cilibrasi, M. Li, Normalized information distance, in: Information theory and statistical learning, Springer, 2009, pp. 45–82.

[26] T.-C. Chen, T. Stepan, S. Dick, J. Miller, An Anti-Phishing system employing diffused information, ACM Trans. Inf. Syst. Secur. 16 (4) (2014) 16:1–16:31.

[27] M. O'Neill, C. Ryan, Genetic code degeneracy: Implications for grammatical evolution and beyond, in: European Conference on Artificial Life, Springer, 1999, pp. 149–153.

[28] M. O'Neill, C. Ryan, Under the hood of grammatical evolution, in: Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation-Volume 2, Morgan Kaufmann Publishers Inc., 1999, pp. 1143–1148.

[29] M. O'Neill, C. Ryan, Grammatical Evolution, Springer Nature, 2003.

[30] P. Nordin, W. Banzhaf, F. D. Francone, Introns in nature and in simulated structure evolution, in: Bio-Computation and Emergent Computation, Skovde, Sweden, 1997, pp. 1–14.

[31] N. J. Radcliffe, P. D. Surry, Fitness Variance of Formae and Performance Prediction, in: FOGA, Vol. 3, 1994, pp. 51–72.

[32] F. Rothlauf, D. E. Goldberg, Redundant representations in evolutionary computation, Evolutionary Computation 11 (4) (2003) 381–415.

[33] F. Rothlauf, M. Oetzel, On the locality of grammatical evolution, in: European Conference on Genetic Programming, Springer, 2006, pp. 320–330.

[34] D. Wilson, D. Kaur, Search, neutral evolution, and mapping in evolutionary computing: A case study of grammatical evolution, IEEE Transactions on Evolutionary Computation 13 (3) (2009) 566–590.

[35] A. Thorhauer, On the Non-uniform Redundancy in Grammatical Evolution, in: International Conference on Parallel Problem Solving from Nature, Springer, 2016, pp. 292–302.

[36] I. Dempsey, M. O'Neill, A. Brabazon, Foundations in grammatical evolution for dynamic environments, Vol. 194, Springer, 2009.

[37] M. Črepinšek, S.-H. Liu, M. Mernik, Exploration and exploitation in evolutionary algorithms: A survey, ACM Computing Surveys (CSUR) 45 (3) (2013) 35.

[38] T. Back, Selective pressure in evolutionary algorithms: A characterization of selection mechanisms, in: Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on, Citeseer, 1994, pp. 57–62.

[39] E. Medvet, A. Bartoli, A. De Lorenzo, F. Tarlao, GOMGE: Gene-Pool Optimal Mixing on Grammatical Evolution, in: International Conference on Parallel Problem Solving from Nature, Springer, 2018, pp. 223–235.

[40] M. Nicolau, Understanding grammatical evolution: initialisation, Genetic Programming and Evolvable Machines 18 (4) (2017) 467–507.

[41] M. Nicolau, A. Agapitos, Understanding grammatical evolution: Grammar design, in: Handbook of Grammatical Evolution, Springer, 2018, pp. 23–53.

[42] P. A. Whigham, Grammatically-based genetic programming, in: Proceedings of the workshop on genetic programming: from theory to real-world applications, 1995, pp. 33–41.

[43] N. Lourenço, F. B. Pereira, E. Costa, SGE: a structured representation for grammatical evolution, in: International Conference on Artificial Evolution (Evolution Artificielle), Springer, 2015, pp. 136–148.

Table 7: Error on validation data of the best individual: mean and standard deviation across runs.

	Strategy	Keijzer6	Pagie1
CFGGP	SSG3P	39.6 ±14.5	12.1 ± 6.8
	PPG3P-P-U	36.8 ±14.9	8.9 ± 5.2
	PPG3P-P-A$^-$	38.6 ±12.7	**6.7** ± 5.1
	PPG3P-P-A$^+$	40.4 ± 9	8.7 ± 4
	PPG3P-F-U	38.6 ±10.5	15.3 ± 7.7
	PPG3P-F-A$^-$	41.6 ±21.9	10.3 ± 3.7
	PPG3P-F-A$^+$	40.6 ±13.6	10.1 ± 5
	PPG3P-F-L$^-$	39.3 ±12.7	9 ± 4.1
	PPG3P-F-L$^+$	34 ±14.6	8.5 ± 7.1
	DCG3P-P	32.4 ±16.8	9.2 ± 4.1
	DCG3P-F	**22.1** ±10.2	10.1 ± 6.3
GE	SSG3P	35.4 ±25.8	23.7 ± 8.4
	PPG3P-G-U	54.3 ±18.5	16.7 ± 7.5
	PPG3P-P-U	39.5 ±11.2	19.7 ± 6.3
	PPG3P-P-A$^-$	41.4 ±16.7	20.6 ± 7.5
	PPG3P-P-A$^+$	35.9 ±21.2	19.4 ± 6.1
	PPG3P-F-U	38.2 ±13.5	17.5 ± 5.4
	PPG3P-F-A$^-$	47.3 ±25.5	18.6 ± 4.2
	PPG3P-F-A$^+$	42.1 ±20.9	19.9 ± 6
	PPG3P-F-L$^-$	46.8 ±25.8	14.8 ± 6.6
	PPG3P-F-L$^+$	**26.4** ±10	17.4 ± 9.2
	DCG3P-G	43.1 ±11.4	19.3 ± 3.4
	DCG3P-P	44.4 ±11.1	17 ± 8.2
	DCG3P-F	40 ±20.6	**13.8** ± 4.3
SGE	SSG3P	41.4 ±37.8	20.4 ± 7.6
	PPG3P-G-U	38.8 ±46.5	23.1 ± 9.9
	PPG3P-P-U	**13.3** ± 0	14.8 ± 3.1
	PPG3P-P-A$^-$	14 ± 4.4	16.4 ± 3.4
	PPG3P-P-A$^+$	13.9 ± 3.9	16 ± 4.3
	PPG3P-F-U	14.4 ± 4.9	**13.1** ± 2.9
	PPG3P-F-A$^-$	15.8 ± 7.6	16.3 ± 2.9
	PPG3P-F-A$^+$	**13.3** ± 0	13.8 ± 3.8
	PPG3P-F-L$^-$	18.2 ± 9.8	14.3 ± 3.7
	PPG3P-F-L$^+$	15.6 ± 7	16.7 ± 2.7
	DCG3P-G	14.6 ± 7.9	15.7 ± 3.7
	DCG3P-P	**13.3** ± 0	15 ± 3.7
	DCG3P-F	13.7 ± 2.4	15.2 ± 4.3
WHGE	SSG3P	46.7 ± 5.6	16.8 ±24.6
	PPG3P-G-U	35.4 ±15.1	**11.1** ± 3.3
	PPG3P-P-U	42.8 ±10.3	14.8 ± 5.8
	PPG3P-P-A$^-$	47 ±14.6	11.3 ± 4.2
	PPG3P-P-A$^+$	**31.5** ±18.1	11.8 ± 7.6
	PPG3P-F-U	35.3 ±14.3	54.2 ±65.2
	PPG3P-F-A$^-$	40.2 ± 9.9	14.3 ± 3
	PPG3P-F-A$^+$	37.4 ±11.8	11.7 ± 4.9
	PPG3P-F-L$^-$	41.9 ± 7.3	11.5 ± 4.9
	PPG3P-F-L$^+$	35.2 ±11	14.1 ± 4.8
	DCG3P-G	37.3 ±13.1	14.8 ± 6.8
	DCG3P-P	37.5 ±13.2	11.9 ± 2.5
	DCG3P-F	42 ±10.8	12.6 ± 4.4

[44] A. Bartoli, M. Castelli, E. Medvet, Weighted hierarchical grammatical evolution, IEEE transactions on cybernetics (2018) 1–13doi:10.1109/TCYB.2018.2876563.

[45] S. Chiam, C. Goh, K. Tan, Issues of Binary Representation in Evolutionary Algorithms, in: 2006 IEEE Conference on Cybernetics and Intelligent Systems, IEEE, 2006, pp. 1–8. doi:10.1109/iccis.2006.252329.
URL https://doi.org/10.1109/iccis.2006.252329

[46] C. Ryan, M. O'Neill, Grammatical evolution: A steady state approach, Late Breaking Papers, Genetic Programming 1998 (1998) 180–185.

[47] K. A. De Jong, Evolutionary computation: a unified approach, MIT press, 2006.

[48] S. Luke, Essentials of metaheuristics, Vol. 113, Lulu Raleigh, 2009.

[49] J. McDermott, D. R. White, S. Luke, L. Manzoni, M. Castelli, L. Vanneschi, W. Jaskowski, K. Krawiec, R. Harper, K. De Jong, et al., Genetic programming needs better benchmarks, in: Proceedings of the 14th annual conference on Genetic and evolutionary computation, ACM, 2012, pp. 791–798.

[50] D. R. White, J. Mcdermott, M. Castelli, L. Manzoni, B. W. Goldman, G. Kronberger, W. Jaśkowski, U.-M. O'Reilly, S. Luke, Better GP benchmarks: community survey results and proposals, Genetic Programming and Evolvable Machines 14 (1) (2013) 3–29.

[51] L. Vanneschi, M. Castelli, L. Manzoni, The K landscapes: a tunably difficult benchmark for genetic programming, in: Proceedings of the 13th annual conference on Genetic and evolutionary computation, ACM, 2011, pp. 1467–1474.

[52] M. Keijzer, Improving symbolic regression with interval arithmetic and linear scaling, Genetic programming (2003) 275–299.

[53] N. Q. Uy, N. X. Hoai, M. O'Neill, R. I. McKay, E. Galván-López, Semantically-based crossover in genetic programming: application to real-valued symbolic regression, Genetic Programming and Evolvable Machines 12 (2) (2011) 91–119.

[54] L. Pagie, P. Hogeweg, Evolutionary consequences of coevolving targets, Evolutionary computation 5 (4) (1997) 401–418.

[55] E. Medvet, A. Bartoli, A. Ansuini, F. Tarlao, Observing the Population Dynamics in GE by means of the Intrinsic Dimension, CoRR abs/1812.02504 (2018). arXiv:1812.02504.
URL http://arxiv.org/abs/1812.02504

[56] E. Medvet, A. Bartoli, A. De Lorenzo, F. Tarlao, Designing automatically a representation for grammatical evolution, Genetic Programming and Evolvable Machines (Jul 2018). doi:10.1007/s10710-018-9327-2.