GOMGE: Gene-pool Optimal Mixing on Grammatical Evolution

Eric Medvet, Alberto Bartoli, Andrea De Lorenzo, and Fabiano Tarlao

Department of Engineering and Architecture, University of Trieste, Trieste, Italy

Abstract. Gene-pool Optimal Mixing Evolutionary Algorithm (GOMEA) is a recent Evolutionary Algorithm (EA) in which the interactions among parts of the solution (i.e., the linkage) are learned and exploited in a novel variation operator. We present GOMGE, the extension of GOMEA to Grammatical Evolution (GE), a popular EA based on an indirect representation which may be applied to any problem whose solutions can be described using a context-free grammar (CFG). GE is a general approach that does not require the user to tune the internals of the EA to fit the problem at hand: there is hence the opportunity for benefiting from the potential of GOMEA to automatically learn and exploit the linkage. We apply the proposed approach to three variants of GE differing in the representation (original GE, SGE, and WHGE) and incorporate in GOMGE two specific improvements aimed at coping with the high degeneracy of those representations. We experimentally assess GOMGE and show that, when coupled with WHGE and SGE, it is clearly beneficial to both effectiveness and efficiency, whereas it delivers mixed results with the original GE.

Keywords: Genetic Programming, Linkage, Family of Subsets, Representation

1 Introduction

Evolutionary Algorithms (EAs) are a powerful tool for solving complex problems. One motivation for their wide adoption is that the user is not required to provide a model for the problem at hand: in most cases, it is up to the EA to figure out how the parts of the solution (w.r.t. the representation employed in that EA) interact in determining the solution quality. However, actually knowing the model and being able to exploit its knowledge may be crucial to determine the effectiveness of the EA.

A model-based EA has been recently proposed for achieving both goals, i.e., the ability to know and exploit the model without requiring any user-provided specification of the model itself. The Gene-pool Optimal Mixing Evolution Algorithm (GOMEA) [1] is a state-of-the-art approach for solving discrete optimization problems and has been carefully designed for exploiting the interactions among parts of a solution, i.e., the *linkage*. GOMEA is based on several crucial contributions: an internal representation of the linkage; a method for deriving the linkage from the population; a novel genetic operator (Gene-pool Optimal Mixing, GOM) in which the individual iteratively receives from random donors some portions of the genetic material defined by the linkage.

We here present GOMGE, i.e., the extension of GOMEA to Grammatical Evolution (GE) [2]—a form of Genetic Programming (GP). GE particularly fits the two aforementioned goals pursued by GOMEA, because it is really a general purpose EA. In facts, key for GE success is that it can tackle any problem whose solutions may be described by means of a context-free grammar (CFG). The user is hence not required to know and tune the internals of the EA: he can obtain a solution for the problem at hand by simply providing the CFG and a fitness function. Indeed, GE has been widely used in many different applications: e.g., generation of string similarity indexes suitable for text extraction [3], road traffic rules synthesis [4], automatic design of analog electronic circuits [5], and even the design of other optimization algorithms [6].

Internally, GE operates on individuals described with an indirect representation: genetic operators are applied to bit-string genotypes; then, bit-strings are transformed into solutions (i.e., strings of the language defined by the problem-specific CFG) by means of a genotype-phenotype mapping function. The latter, which essentially defines the individual representation of GE, favored the adoption of this EA, since it allowed building on the vast knowledge about manipulation of bit-string genotypes. On the other hand, extensive research on the properties of GE representation showed that it has many drawbacks [7–9]. Indeed, beyond inspiring a large debate among scholars which also concerned about the aims and methods for designing an EA representation [10–12], the drawbacks of GE representation also stimulated the recent arising of two variants—Structured GE (SGE) [13] and Weighted Hierarchical GE (WHGE) [14]—mainly consisting in a different genotype-phenotype mapping function and, hence, a different representation.

We applied GOMGE to the three mentioned variants of GE (the original GE, SGE, and WHGE) and incorporated in GOMGE two small modifications motivated by the need of coping with the degeneracy of those representations, i.e., the tendency to map many genotypes to the same phenotype [7, 8]. Our work has a twofold aim: (a) extend the benefit in effectiveness delivered by GOMEA to GE, hence further boosting its practical applicability, and (b) shed new lights on the three representations, in particular concerning their proneness to exhibit "good" linkage, i.e., a linkage which can actually be exploited to improve the effectiveness of the EA. The latter point is of particular interest for better understanding both GOMEA (and its linkage learning method) and GE representations: in facts, being based on an indirect representation, the linkage observed in GE is the result of the combination of interactions between genes which occur during the genotype-phenotype mapping and those related to the problem at hand.

We performed an extensive experimental evaluation considering three GE variants with four linkage models applied to four benchmark problems. The results show that GOMGE does improve the effectiveness and efficiency of both SGE and WHGE, whereas it delivers mixed results with GE.

The remainder of the paper is organized as follows. In Section 2, we briefly survey previous studies relevant to the present paper. In Sections 3 and 3.1, we describe the standard search algorithm used in GE and GOMGE, respectively. In Section 4, we discuss the results of our experimental evaluation. Finally, in Section 5, we summarize the findings and draw the conclusions.

2 Related works

GOMEA [1] has been extended to different EAs or macro-categories of problems: to GP in GOMEA-GP [15], where a novel approach is proposed for identifying and encapsulating relevant building blocks; to real-valued (RV) optimization in RV-GOMEA [16]; to multi-objective (MO) optimization problems in MO-GOMEA [17]. GOMGE is the first application of GOMEA to GE.

There have been several attempts to exploit some form of knowledge of the model underlying the problem for improving the effectiveness of GE. These efforts were usually aimed at discovering useful *building blocks*, i.e., reusable components of the solutions. Position-independent GE (πGE) [18] proposed a novel genotype-phenotype mapping in which the non-terminal symbol to be derived was chosen using the information in the genotype instead of following a left-to-right order. Decoupling non-terminal derivation from the non-terminal choice was expected to favor the emergence of building blocks, but no experimental evidence of the desired effect was provided. A different approach was instead proposed in [19] and, more recently, in [20]. In both cases, the aim is to modify the grammar to discover new problem-specific building blocks and hence improve the search effectiveness; the two cited papers, however, greatly differ in the way they pursue this goal. In [19] a user-defined "universal grammar" related to the class of considered problems (e.g., symbolic regression) is available and part of the genotype is devoted to encode a more specific grammar which describes the actual solution space. In [20], a two phases process is proposed: in a first phase, a probabilistic grammar-based model is learned during an evolution performed using the original user-provided CFG. In a second phase, the new learned model is used to evolve hopefully better solutions. In the present work, differently than the two cited works, we attempt to learn the model (i.e., the linkage) directly at the level of the genotype, instead of at the level of the phenotype (i.e., the grammar).

Another attempt of incorporating the knowledge of the model in GE has been proposed in [21] and further improved in [22]. The authors of the cited paper describe a rather complex theoretical framework in which a model can be obtained from a grammar by means of a deterministic algorithm: the model is a particular graph in which vertexes are partially derived strings of the language and edges are derivation rules. The genotype is not a bit-string, but instead a sequence of derivation rules which, essentially, allows to move along the graph. Accordingly, the proposed EA uses specific genetic operators, making it rather different than the original GE. Finally, it is worth to mention that in several studies of model-based GP, the proposed model itself consisted in (probabilistic) grammars (e.g., [23]): we refer the reader to [24] for a broader overview of these approaches.

3 Grammatical Evolution

GE has been widely studied and several variants for the various EA components have been proposed. Here, we present the most widely used search algorithm and representation (for which we consider also the recent variants SGE and WHGE), because they are relevant to this study.

Algorithm 1 shows the search algorithm of GE. First, the initial population I consisting of n_{pop} individuals is built. In this work, we consider an initialization procedure in which genotypes of a given length l_g are generated at random, but more complex strategies may be employed. Then, the following steps are repeated until the termination criterion is met: (1) A new population I' (with $|I'| = |I| = n_{\text{pop}}$) is built from I by applying the genetic operators (mutation and crossover chosen according to the probabilities $p_{\text{mut}}, p_{\text{cross}}$) to parents selected from the population I using a predefined parent selection criterion (SelectParent() in Algorithm 1). (2) The population I is updated by including the new population I' and then by removing the n_{pop} exceeding individuals using a predefined removal selection criterion (SelectRemoval() in Algorithm 1).

Concerning the termination criterion, the most common option is to repeat the two steps above for a predefined number of times, usually called the number of generations. In this work, however, we chose a different criterion for enabling a fairer comparison with GOMGE which, differently than the $m + n$ generational model of Algorithm 1, may perform a large number of fitness evaluations in each iteration of the main loop (see Section 3.1). We hence adopted for both search algorithms the following stopping criterion. The steps are iterated until at least one of the two following conditions is met: (a) the elapsed time T after the beginning of the evolution exceeds a predefined time limit $T_{\max}$ or (b) the population I includes an individual with perfect fitness f (the notion of perfect fitness being dependent, in general, on the problem).

The search algorithm defined in Algorithm 1 is agnostic to the specific selection criteria SelectParent() and SelectRemoval(): tournament selection and worst fitness selection (i.e., truncation selection) are, respectively, common choices. The genotype-phenotype mapping function Map() is the component in which the GE variants mostly differ and essentially defines the individual representation. In this work, we consider the original representation and two recent variants: Structured GE (SGE) [13] and Weighted Hierarchical GE (WHGE) [14]: it is worth to note that, in both cases, the proposal of the representation variant was aimed at improving the poor properties of the original GE representation, in particular degeneracy and locality. Degeneracy concerns the degree to which different genotypes are mapped to the same phenotype. Locality describes how well genotypic neighbors correspond to phenotypic neighbors. It has been shown that these properties are related to higher level properties of the EA, as, e.g., diversity [7] and evolvability [25]. It is foreseeable that degeneracy and locality may impact also on the learnability of the linkage and may interplay with the GOM operator.

Algorithm 1 Standard GE.

```
function Evolve()
  $I \leftarrow$ InitPopulation($n_{\text{pop}}$)
  while $\neg$TerminationCriterionMet() do
    $I' \leftarrow \emptyset$
    while $|I'| < n_{\text{pop}}$ do
      $o \leftarrow$ GetOperator()
      $G_p \leftarrow \emptyset$
      while $|G_p| \leq$ Arity($o$) do
        $(g_p, p_p, f_p) \leftarrow$ SelectParent($I$)
        $G_p \leftarrow G_p \cup \{g_p\}$
      end while
      $G_c \leftarrow$ Apply($o, G_p$)
      for all $g_c \in G_c$ do
        $p_c \leftarrow$ Map($g_c$)
        $f_c \leftarrow$ Fitness($p_c$)
        $I' \leftarrow I' \cup \{(g_c, p_c, f_c)\}$
      end for
    end while
    $I \leftarrow I \cup I'$
    while $|I| > n_{\text{pop}}$ do
      $I \leftarrow I \setminus \{$SelectRemoval($I$)$\}$
    end while
  end while
  return Best($I$)
end function
```

Algorithm 2 GOMGE.

```
function Evolve()
  $I \leftarrow$ InitPopulation($n_{\text{pop}}$)
  $i^\star =$ Best($I$)
  while $\neg$TerminationCriterionMet() do
    $\mathcal{F} \leftarrow$ LearnLinkageModel($I$)
    $I' \leftarrow \emptyset$
    for all $(g, p, f) \in I$ do
      $(g_0, p_0, f_0) \leftarrow (g, p, f)$
      for all $F \in$ RndPerm($\mathcal{F}$) do
        $g_c \leftarrow g$
        $(g_d, p_d, f_d) \leftarrow$ RandomDonor($I$)
        $g_c[F] \leftarrow g_d[F]$
        $p_c \leftarrow$ Map($g_c$)
        $f_c \leftarrow$ Fitness($p_c$)
        if $f_c < f$ then
          $(g, p, f) \leftarrow (g_c, p_c, f_c)$
        end if
      end for
      while $p = p_0$ do
        $g \leftarrow$ Apply(Mutation, $\{g\}$)
        $p \leftarrow$ Map($g$)
        $f \leftarrow$ Fitness($p$)
      end while
      $I' \leftarrow I' \cup \{(g, p, f)\}$
    end for
    $I \leftarrow I'$
    $i^\star =$ Best($I \cup \{i^\star\}$)
  end while
  return $i^\star$
end function
```

Due to space constraints, we do not describe in details the representation of GE, SGE, and WHGE: we provide a coarse overview of the underlying principles and refer the reader to the respective papers for further details. Being forms of grammar-based genetic programming, in GE, SGE, and WHGE the phenotype is a string of the language language $\mathcal{L}(\mathcal{G})$ defined by a user-provided CFG $\mathcal{G}$, which is an implicit parameter of the mapping function Map(). In the original GE [2], the genotype g is a bit-string. Groups of 8 consecutive bits in the genotype are called codons: each *codon* encodes an integer value and is consumed for deriving the leftmost non-terminal. SGE has been introduced in [13] by Lourenço et al. In SGE, the genotype g consists of a number of fixed-size lists (*genes*) of integers: each list corresponds to a non-terminal symbol of the CFG and each integer in the list (*codon*) determines a single derivation for that non-terminal. Finally, the most recent WHGE [14] is designed to consume the genotype hierarchically with the aim of reducing the degeneracy and increasing the locality. In WHGE, the genotype g is a bit-string, as in the original GE.

3.1 GOMGE: Gene-pool Optimal Mixing EA for GE

Our GOMGE proposal consists on two localized modifications to the adaptation of GOMEA to GP [15], described below and motivated by explorative experiments and recent findings about (lack of) diversity in GE [26, 7]. GOMGE is described in Algorithm 2. After the initialization of the population, the main loop is repeated until a termination criterion is met and consists in two steps: (i) learning the linkage from the current population and (ii) applying the Gene-pool Optimal Mixing (GOM) variation operator to each individual in the population. The linkage is expressed as a Family of Subsets (FOS) $\mathcal{F} = \{F_1, F_2, \dots\}$ which is a set of sets of zero-based genotype indexes (*loci*): i.e., $F_i \subseteq, \{0, \dots, l_g - 1\}$, where l_g is the evolution-wise immutable size of the genotype. We experimented with 4 different way of obtaining the FOS described at the end of this section.

Applying the GOM operator to an individual (g, p, f) consists in repeating the following steps for each set F in a random permutation of $\mathcal{F}$: (i) a donor (g_d, p_d, f_d) is randomly chosen in the population and (ii) the portions of the genotype g defined by F are replaced with the corresponding portions coming from g_d; (iii) the fitness $\textsc{Fitness}(\textsc{Map}(g))$ of the new individual is computed and, (iv) if there is a strict improvement, the modification on the individual g is kept, otherwise, it is rolled back.

After preliminary experiments, we observed that this version of the GOM often resulted in no modifications being applied to the individual, since no fitness improvements were obtained. We think this finding is motivated by the degeneracy of the indirect representation of GE and its variants: the likelihood is non-negligible of obtaining the same individual after an iteration of GOM operator; as a consequence, the fitness does not improve and the evolution might stagnate. We hence modified the GOM operator by employing a *forced mutation* (performed with a mutation operator suitable to the specific representation being used) in case the phenotype did not change after the processing of all the sets in $\mathcal{F}$. The idea is borrowed from [15], where a phase called "forced improvement" eventually results in an individual with a better fitness than the input one, yet possibly equal to another individual in the population. Here, we instead simply apply a standard mutation, because otherwise the tendency of GE and variants to drastically reduce the diversity in the population during the evolution could have been further stimulated.

Since the forced mutation might apply to the best individual in the population (hence negatively affecting the results of the search obtained so far), we introduced a simple mechanism for keeping track of the best individual $i^\star$, which is updated at each iteration of the main loop.

In GOMGE, as in GOMEA, the linkage is expressed using a FOS, which can either be learned from the population or being predefined. We considered 4 variants, two belonging to the former category (Linkage Tree and Random Tree) and two to the latter category (Univariate and Natural).

The Univariate FOS (later denoted by U) is the simplest FOS and assumes that there is no linkage between portions of the genotype. This FOS contains one singleton set for each possible locus: $\mathcal{F}_\text{U} = \{\{0\}, \{1\}, \dots, \{l_g - 1\}\}$.

The Natural FOS (later denoted by N) is a statically built FOS which tries to capture the representation-dependent linkage. We defined it only for GE, where it captures the fact that derivations are chosen using groups of 8 consecutive bits (i.e., $\mathcal{F}_{\text{N,GE}} = \{\{0, 1, \dots, 7\}, \{8, 9, \dots, 15\}, \dots\}$), and for SGE, where it captures the fact that integers are organized in lists, the size of each list being dependent on the grammar (i.e., $\mathcal{F}_{\text{N,SGE}} = \{\{0, \dots, |g_{s_0}| - 1\}, \{|g_{s_0}|, \dots, |g_{s_0}| + |g_{s_1}| - 1\}, \dots\}$).

The learnable variants are Linkage Tree and Random Tree, later denoted by LT and RT, respectively. LT was already considered in the seminal GOMEA paper and was later shown to be very beneficial to search effectiveness, in particular in black-box optimization problems [27]. LT models complex linkage structures using a hierarchy, i.e., a tree where nodes are sets of loci and a node is the set union of its children. The LT FOS $\mathcal{F}_{\text{LT}}$ is built from the population as follows. Initially, a set of sets of loci $\mathcal{F}_0$ is set to $\mathcal{F}_{\text{U}}$ and $\mathcal{F}_{\text{LT}} = \mathcal{F}_0$. Then, the following steps are repeated until $|\mathcal{F}_0| \geq 1$: (i) the pair $F, F' \in \mathcal{F}_0 \times \mathcal{F}_0$ of loci sets, with $F \neq F'$, is determined which exhibits the greatest mutual information; then (ii) F, F' are removed from $\mathcal{F}_0$ and $F \cup F'$ is added to $\mathcal{F}_0$ and $\mathcal{F}_{\text{LT}}$. This procedure may be implemented efficiently using the algorithm described in [28], where the mutual information between sets of loci is estimated, rather than computed using the genotype values at F, F' loci observed in the population (we refer the reader to the cited paper for further details).

Finally, RT resembles LT since it also models the linkage as a hierarchy. Differently than LT, however, a random value instead of the mutual information is used at step i above when building $\mathcal{F}_{\text{RT}}$. The rationale is to allow, as for LT, the simultaneous modification at different loci of the genotype.

4 Experimental evaluation

For assessing experimentally the effectiveness of GOMGE w.r.t. the standard GE search algorithm, we considered 4 benchmark problems: Parity (with $n \in \{5, \dots, 9\}$), Nguyen7 [29], KLandscapes [30] (with $k \in \{3, \dots, 7\}$), and Text [7]. These problems represent different domains, including boolean functions (Parity), symbolic regression (Nguyen7), and synthetic problems (KLandscapes and Text). Two of them have a tunable hardness (Parity and KLandscapes) and two are recommended as GP benchmarks in [31] (Nguyen7 and KLandscapes); one (Text) has been designed purposely for grammar-based GP and is based on a grammar with more derivation rules and more symbols than the other considered problems.

We performed the experimental evaluation using a prototype Java implementation of both standard GE and GOMGE. The implementation and the grammars for the benchmark problems are publicly available[1]. The prototype includes a two caches for the fitness function Fitness() and the genotype-phenotype mapping function Map(); both use a size-based eviction policy with a size limit of 200 000 entries.

We performed 30 runs for each of the five variants (standard GE, to be considered as the *baseline* and later denoted by Base., and GOMGE coupled with

[1] `https://github.com/ericmedvet/evolved-ge`

the 4 FOSs, U, N, RT, and LT) on each of the four problems. We executed each run on one node of the CINECA HPC cluster (Marconi-A1), the node having 2 Intel Xeon E5-2694 v4 CPUs (2.3 GHz) with 18 cores each and 128 GB of RAM.

We set the main evolutionary parameters as follows: genotype size $l_g = 512$ for GE, $l_g = 128$ for WHGE, or determined by $d = 6$ (see [13]) for SGE; population size $n_{\text{pop}} = 500$; two-points same crossover for GE, WHGE or SGE crossover for SGE with rate 0.8; bit-flip mutation with $p_{\text{mut}} = 0.01$ for GE, WHGE or SGE mutation with $p_{\text{mut}} = 0.01$ for SGE with rate 0.2; tournament selection of size 3; and max elapsed time $T_{\max} = 60\,\text{s}$.

Table 1 shows the mean and the standard deviation (across the 30 runs) of the final best fitness for each problem and variant. The table also shows, graphically and for each GOMGE variant and problem, the statistical significance (p-value with the Mann-Whitney U-test) of the null hypothesis that the final best fitness values have equal median of those obtained with the baseline.

Table 1. Mean and standard deviation of the final best fitness. The best mean for each problem is highlighted. The statistical significance (see text) is shown graphically: ‡ means $p < 0.01$, † means $p < 0.05$, and * means $p < 0.1$ (no markers for greater p-values).

	Var.	Parity-7	Nguyen7	KLand.-5	Text
GE	Base.	0.5 ±0.02	0.39±0.25	0.61±0.09	4.9±1.2
GE	U	0.5 ±0.01	0.49±0.19‡	0.63±0.06‡	3.5±0.7‡
GE	N	0.5 ±0	0.4 ±0.2	0.61±0.09	**3.1**±0.8‡
GE	RT	0.49±0.02	0.68±0.6 ‡	0.68±0.04‡	4.5±0.6‡
GE	LT	0.49±0.03	0.68±0.16‡	0.68±0.04‡	4.6±0.5‡
WHGE	Base.	0.17±0.13	0.52±0.19	0.4 ±0.08	5.7±0.8
WHGE	U	0.16±0.07*	0.31±0.15‡	0.6 ±0.05‡	4.9±0.5‡
WHGE	RT	**0** ±0 ‡	**0.18**±0.11‡	0.29±0.04‡	4 ±0 ‡
WHGE	LT	**0** ±0 ‡	0.21±0.12‡	**0.25**±0.07‡	4 ±0 ‡
SGE	Base.	0.08±0.12	0.7 ±0.12	0.54±0.14	6.3±0.5
SGE	U	**0** ±0 ‡	0.35±0.23‡	0.34±0 ‡	5.4±0.5‡
SGE	N	**0** ±0 ‡	0.29±0.24‡	0.34±0 ‡	5.1±0.3‡
SGE	RT	**0** ±0 ‡	0.65±1.14‡	0.34±0 ‡	5 ±0.2‡
SGE	LT	**0** ±0 ‡	0.54±0.21‡	0.34±0 ‡	5 ±0.2‡

The foremost finding is that GOMGE outperforms the baseline with WHGE and SGE in almost all cases (i.e., FOS and problem), with the single exception of U with WHGE on the KLandscapes-5 problem, for which the baseline performs better (0.4 vs. 0.6). The difference is always significant. GOMGE improvement becomes evident on the Parity-7 problem, for which both WHGE and SGE obtain the perfect fitness in all the runs only with GOMGE. Concerning the FOS, it can be seen that the largest improvement is delivered by RT and LT, for WHGE, and by N, for SGE (see also the next tables): the facts that LT and RT lead to the same good performance with WHGE and that N with SGE resembles the SGE crossover operator (see [13]) suggest that the improvement is related to the reiterated application of the GOM operator, rather than to the possibility of learning the linkage.

Differently, coupling GOMGE with GE representation leads to mixed results: no significant difference are visible on one problem (Parity-7), a decrease in the fitness is visible on two problems (Nguyen7 and KLandscapes-5), and an improvement is visible for the last problem (Text). Overall, N is the best FOS for GE.

For better understanding the results in terms of final best fitness, we analyzed also three other relevant metrics: the elapsed time T, the number of actual fitness evaluations N (corresponding to the fitness cache miss count), and the final phenotypical diversity D. We measured the latter as the ratio between the number of different phenotypes in the population and the population size.

Table 2. Elapsed time T (in s), number N of actual fitness evaluations (in thousands), and final phenotype diversity D (in percentage).

		Elapsed time T [s]				Act. fitness ev. N [$\times 10^3$]				Pheno. div. D [%]			
	Var.	Par.-7	Ng.7	KL.-5	Text	Par.-7	Ng.7	KL.-5	Text	Par.-7	Ng.7	KL.-5	Text
GE	Base.	85	66	65	59	0.2	6.8	5	5	6	4	7	2
	U	114	57	59	59	0.4	12.8	19.2	192.8	23	35	68	93
	N	123	59	55	60	0.2	37.8	32.1	133.7	28	54	69	97
	RT	145	65	65	68	0.7	7.9	14.7	67.9	27	53	69	96
	LT	149	79	72	69	0.6	8	14.6	53.1	28	54	69	96
WHGE	Base.	68	64	64	61	10.7	9.4	10.5	6.5	11	4	31	12
	U	71	64	71	62	585.2	299.8	160.2	197.6	92	89	88	92
	RT	14	86	61	77	353.7	548.7	352.7	591.9	79	56	89	48
	LT	15	85	61	83	328.7	360.7	321.4	457.1	71	45	54	51
SGE	Base.	43	61	62	62	1.2	3.3	2.6	1.2	5	7	5	4
	U	20	60	64	60	39.6	50.1	52.1	45.1	98	71	62	91
	N	1	63	63	59	38.8	89.9	53.3	62	98	71	38	82
	RT	2	57	76	58	47.3	81.2	54.8	46.3	92	52	27	50
	LT	4	53	68	59	57.9	21.8	52	33.4	64	27	27	32

Table 2 shows the mean (across the 30 runs) of the three metrics T, N, D for each problem and variant. Two main observations may be made. First, the number of actual fitness evaluation increases with GOMGE, the increment being remarkable for WHGE and SGE (up to $20\times$). This figure is also reflected in the elapsed time T, when a perfect fitness value is not found. We recall that one of the two termination criteria is the elapsed time, with a time limit of $T_{\max} = 60\,\mathrm{s}$: however, since the condition is evaluated once per main loop, the limit may be exceeded, in particular for GOMGE. It can also be noted that GE, in all the 5 variants, performs a very low number (hundreds, on average) of actual fitness evaluations for the Parity-7 problem: this is mainly due to high degeneracy, which has already been shown to hamper this representation in particular in problems where the phenotype should be large [7], and, to a lesser extent, to invalidity, i.e., the tendency of generating a null phenotype after exceeding the maximum number of wrappings [2].

Second, Table 2 shows that the final phenotypical diversity is in general much larger with GOMGE than with the baseline, in all cases: values are around 10% with the latter and often exceed 90% with GOMGE. This finding can be explained by the fact that GOMGE does not select best individuals for applying the GOM operator, but rather replaces a parent with a child only upon a fitness improvement, a mechanism resembling the established diversity promotion scheme known as deterministic crowding [32]. Together, the two observations suggest that GOMGE is at the same time more effective and more efficient than the baseline in exploring the search space: indeed, it can be noted from Tables 1 and 2 than the largest fitness improvements are obtained in sync with improvements in the metrics N and D. Significantly, the only problem in which GOMGE outperforms the baseline with GE is the one (Text) in which the increment of N and D is the greatest.

5 Concluding remarks

We presented GOMGE, an application of the Gene-pool Optimal Mixing Evolutionary Algorithm (GOMEA) to Grammatical Evolution (GE), a form of general-purpose Genetic Programming which is widely used by practitioners because it easily applies to any problem whose solutions may be described by a context-free grammar. We incorporated in GOMGE two specific improvements for coping with the degeneracy (i.e., the tendency to map many genotypes to the same pohenotype) of GE indirect representations. We applied GOMGE to three variants of GE (original GE, SGE, and WHGE), essentially differing in the individual representation, a key component of any EA which has been shown to impact on many higher-level EA properties (e.g., evolvability), and eventually on its effectiveness.

We performed an extensive experimental evaluation of 4 GOMGE variants, differing in the way of obtaining a linkage model, on 4 benchmark problems. The results show that GOMGE is significantly beneficial to both effectiveness and efficiency of the search with SGE and WHGE, whereas it delivers mixed results with the original GE. At a deeper analysis, the experimental results suggest that the drastic increase in the phenotypical diversity and in the number of actual fitness evaluation are key factors for explaining the performance gap between GOMGE and standard GE.

We think that our proposal further boosts the applicability of GE to practical problems and sheds new light on the possibility of GE representations to exhibit “good” and learnable linkage.

References

1. Thierens, D., Bosman, P.A.: Optimal mixing evolutionary algorithms. In: Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation. GECCO '11, New York, NY, USA, ACM (2011) 617–624
2. Ryan, C., Collins, J., O'Neill, M.: Grammatical evolution: Evolving programs for an arbitrary language. In: European Conference on Genetic Programming, Springer (1998) 83–96
3. Bartoli, A., De Lorenzo, A., Medvet, E., Tarlao, F.: 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, Cham, Springer International Publishing (2016) 260–269
4. Medvet, E., Bartoli, A., Talamini, J.: Road traffic rules synthesis using grammatical evolution. In: EvoApplications (2). (2017) 173–188
5. Castejón, F., Carmona, E.J.: Automatic design of analog electronic circuits using grammatical evolution. Applied Soft Computing **62** (2018) 1003–1018
6. Miranda, P.B., Prudêncio, R.B.: Generation of particle swarm optimization algorithms: An experimental study using grammar-guided genetic programming. Applied Soft Computing (2017)
7. Medvet, E.: A comparative analysis of dynamic locality and redundancy in grammatical evolution. In: European Conference on Genetic Programming, Springer (2017) 326–342
8. Thorhauer, A.: On the non-uniform redundancy in grammatical evolution. In: International Conference on Parallel Problem Solving from Nature, Springer (2016) 292–302
9. Thorhauer, A., Rothlauf, F.: On the locality of standard search operators in grammatical evolution. In: International Conference on Parallel Problem Solving from Nature, Springer (2014) 465–475
10. Medvet, E., Bartoli, A.: On the automatic design of a representation for grammar-based genetic programming. In Castelli, M., Sekanina, L., Zhang, M., Cagnoni, S., García-Sánchez, P., eds.: Genetic Programming, Cham, Springer International Publishing (2018) 101–117
11. Whigham, P.A., Dick, G., Maclaurin, J.: On the mapping of genotype to phenotype in evolutionary algorithms. Genetic Programming and Evolvable Machines (2017) 1–9
12. Spector, L.: Introduction to the peer commentary special section on "on the mapping of genotype to phenotype in evolutionary algorithms" by peter a. whigham, grant dick, and james maclaurin. Genetic Programming and Evolvable Machines **18**(3) (Sep 2017) 351–352
13. Lourenço, N., Pereira, F.B., Costa, E.: Sge: a structured representation for grammatical evolution. In: International Conference on Artificial Evolution (Evolution Artificielle), Springer (2015) 136–148
14. Medvet, E.: Hierarchical grammatical evolution. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion. GECCO '17, New York, NY, USA, ACM (2017) 249–250
15. Virgolin, M., Alderliesten, T., Witteveen, C., Bosman, P.A.N.: Scalable genetic programming by gene-pool optimal mixing and input-space entropy-based building-block learning. In: Proceedings of the Genetic and Evolutionary Computation Conference, ACM (2017) 1041–1048

16. Bouter, A., Alderliesten, T., Witteveen, C., Bosman, P.A.: Exploiting linkage information in real-valued optimization with the real-valued gene-pool optimal mixing evolutionary algorithm. In: Proceedings of the Genetic and Evolutionary Computation Conference, ACM (2017) 705–712
17. Luong, N.H., La Poutré, H., Bosman, P.A.: Multi-objective gene-pool optimal mixing evolutionary algorithms. In: Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation, ACM (2014) 357–364
18. O'Neill, M., Brabazon, A., Nicolau, M., Mc Garraghy, S., Keenan, P.: πgrammatical evolution. In: Genetic and Evolutionary Computation Conference, Springer (2004) 617–629
19. O'Neill, M., Ryan, C.: Grammatical evolution by grammatical evolution: The evolution of grammar and genetic code. Genetic Programming (2004) 138–149
20. Wong, P.K., Wong, M.L., Leung, K.S.: Hierarchical knowledge in self-improving grammar-based genetic programming. In: International Conference on Parallel Problem Solving from Nature, Springer (2016) 270–280
21. He, P., Johnson, C.G., Wang, H.: Modeling grammatical evolution by automaton. Science China Information Sciences **54**(12) (2011) 2544–2553
22. He, P., Deng, Z., Gao, C., Chang, L., Hu, A.: Analyzing grammatical evolution and πgrammatical evolution with grammar model. In: Information Technology and Intelligent Transportation Systems. Springer (2017) 483–489
23. Shan, Y., McKay, R.I., Baxter, R., Abbass, H., Essam, D., Nguyen, H.: Grammar model-based program evolution. In: Evolutionary Computation, 2004. CEC2004. Congress on. Volume 1., IEEE (2004) 478–485
24. Shan, Y., McKay, R., Essam, D., Abbass, H.: A survey of probabilistic model building genetic programming. Scalable Optimization via Probabilistic Modeling (2006) 121–160
25. Medvet, E., Daolio, F., Tagliapietra, D.: Evolvability in grammatical evolution. In: Proceedings of the Genetic and Evolutionary Computation Conference, ACM (2017) 977–984
26. Medvet, E., Bartoli, A., Squillero, G.: An effective diversity promotion mechanism in grammatical evolution. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion, ACM (2017) 247–248
27. Thierens, D., Bosman, P.A.N.: Hierarchical problem solving with the linkage tree genetic algorithm. In: Proceedings of the 15th annual conference on Genetic and evolutionary computation, ACM (2013) 877–884
28. Gronau, I., Moran, S.: Optimal implementations of upgma and other common clustering algorithms. Information Processing Letters **104**(6) (2007) 205–210
29. Uy, N.Q., Hoai, N.X., O'Neill, M., McKay, R.I., Galván-López, E.: Semantically-based crossover in genetic programming: application to real-valued symbolic regression. Genetic Programming and Evolvable Machines **12**(2) (2011) 91–119
30. Vanneschi, L., Castelli, M., Manzoni, L.: The k landscapes: a tunably difficult benchmark for genetic programming. In: Proceedings of the 13th annual conference on Genetic and evolutionary computation, ACM (2011) 1467–1474
31. White, D.R., Mcdermott, J., Castelli, M., Manzoni, L., Goldman, B.W., Kronberger, G., Jaśkowski, W., O'Reilly, U.M., Luke, S.: Better gp benchmarks: community survey results and proposals. Genetic Programming and Evolvable Machines **14**(1) (2013) 3–29
32. Squillero, G., Tonda, A.: Divergence of character and premature convergence: A survey of methodologies for promoting diversity in evolutionary optimization. Information Sciences **329** (2016) 782–799