Geometric Differential Evolution on the Space of Genetic Programs

Alberto Moraglio[1] and Sara Silva[2,3]

[1] School of Computing, University of Kent, Canterbury, UK
a.moraglio@kent.ac.uk
[2] INESC-ID Lisboa, Portugal
[3] Center for Informatics and Systems of the University of Coimbra, Portugal
sara@{kdbio.inesc-id.pt,dei.uc.pt}

Abstract. Geometric Differential Evolution (GDE) is a very recently introduced formal generalization of traditional Differential Evolution (DE) that can be used to derive specific GDE for both continuous and combinatorial spaces retaining the same geometric interpretation of the dynamics of the DE search across representations. In this paper, we derive formally a specific GDE for the space of genetic programs. The result is a Differential Evolution algorithm searching the space of genetic programs by acting directly on their tree representation. We present experimental results for the new algorithm.

1 Introduction

Differential Evolution (DE) is a population-based stochastic global optimization algorithm [16] that has a number of similarities with Particle Swarm Optimization (PSO) and Evolutionary Algorithms (EAs), and has proven to have robust performance over a variety of difficult continuous optimization problems [16]. The search done by DE has a natural geometric interpretation and can be understood as the motion of points in space obtained by linear combinations of their current positions to determine their new positions.

The original formulation of DE requires the search space to be continuous and the points in space to be represented as vectors of real numbers. There are only few extensions of DE to combinatorial spaces [16] [15] [2] [14] and to the space of genetic programs [13]. Some of these works recast the search in discrete spaces as continuous search via encoding the candidate solutions as vectors of real numbers and then applying the traditional DE algorithm to solve these continuous problems. Other works present DE algorithms defined on combinatorial spaces acting directly on the original solution representation that, however, are only loosely related to the traditional DE in that the original geometric interpretation is lost in the transition from continuous to combinatorial spaces. Furthermore, in the latter approaches every time a new solution representation is considered, the DE algorithm needs to be rethought and adapted to the new representation.

GDE [12] is a very recently devised formal generalization of DE that, in principle, can be specified to any solution representation while retaining the original geometric interpretation of the dynamics of the points in space of DE across representations. In particular, GDE can be applied to any search space endowed with a distance and associated with any solution representation to derive formally a specific GDE for the target space and for the target representation. GDE is related to Geometric Particle Swarm Optimization (GPSO) [7], which is a formal generalization of the Particle Swarm Optimization algorithm [3]. Specific GPSOs were derived for different types of continuous spaces and for the Hamming space associated with binary strings [8], for spaces associated with permutations [11] and for spaces associated with genetic programs [17].

In previous work [12], GDE was specialized to the space of binary strings endowed with the Hamming distance and produced good experimental results. In this paper, we extend the study of the GDE algorithm and apply it to searching the space of computer programs represented as parse trees. The main purpose of this paper is to show that this is at all possible, and in particular to show that differential mutation, the core search operator of DE that casts it apart from PSO and EAs, can be readily derived for this non-trivial representation. We also present an initial experimental analysis of this new algorithm, which we call GDE-GP.

The remaining part of the paper is organized as follows. Section 2 describes the general GDE algorithm. Section 3 presents specific GDE search operators for parse trees. Section 4 reports an initial experimental analysis for GDE-GP on standard GP benchmark problems. Section 5 presents the conclusions and future work.

2 Geometric Differential Evolution

In this section, we summarize how the general GDE algorithm was derived (Algorithm 2) [12] from the classic DE algorithm (Algorithm 1). The generalization was obtained using a methodology to generalize search algorithms for continuous spaces to combinatorial spaces [12] based on the geometric framework introduced by Moraglio [6]. The methodology is sketched in the following. Given a search algorithm defined on continuous spaces, one has to recast the definition of the search operators expressing them explicitly in terms of Euclidean distance between parents and offspring. Then one has to substitute the Euclidean distance with a generic metric, obtaining a formal search algorithm generalizing the original algorithm based on the continuous space. Next, one can consider a (discrete) representation and a distance associated with it (a combinatorial space) and use it in the definition of the formal search algorithm to obtain a specific instance of the algorithm for this space. Finally, one can use this geometric and declarative description of the search operator to derive its operational definition in terms of manipulation of the specific underlying representation. This methodology was used to generalize PSO and DE to any metric space, obtaining GPSO and GDE, and then to derive their search operators for specific representations and distances. In particular for DE, the specific GDE for the Hamming space associated with binary strings was derived. In Section 3, we derive the specific

GDE for the space of parse trees with Structural Hamming Distance (SHD) [9] by plugging this distance in the abstract definition of the search operators.

2.1 Classic Differential Evolution

In the following, we describe the traditional DE [16] (see Algorithm 1). The characteristic that sets DE apart from other Evolutionary Algorithms is the presence of the differential mutation operator (see line 5 of Algorithm 1). This operator creates a mutant vector U by perturbing a vector $X3$ picked at random from the current population with the scaled difference of other two randomly selected population vectors $F \cdot (X1 - X2)$. This operation is considered important because it adapts the mutation direction and its step size to the level of convergence and spatial distribution of the current population. The mutant vector is then recombined with the currently considered vector $X(i)$ using discrete recombination and the resulting vector V replaces the current vector in the next population if it has better or equal fitness (in line 7 of Algorithm 1, higher is better).

Algorithm 1 DE with differential mutation and discrete recombination

```
1: initialize population of $N_p$ real vectors at random
2: **while** stop criterion not met **do**
3:   **for all** vector $X(i)$ in the population **do**
4:     pick at random 3 distinct vectors from the current population $X1, X2, X3$
5:     create mutant vector $U = X3 + F \cdot (X1 - X2)$ where $F$ is the scale factor parameter
6:     set $V$ as the result of the discrete recombination of $U$ and $X(i)$ with probability $Cr$
7:     **if** $f(V) \geq f(X(i))$ **then**
8:       set the $i^{th}$ vector in the next population $Y(i) = V$
9:     **else**
10:      set $Y(i) = X(i)$
11:    **end if**
12:  **end for**
13:  **for all** vector $X(i)$ in the population **do**
14:    set $X(i) = Y(i)$
15:  **end for**
16: **end while**
```

2.2 Generalization of Differential Mutation

Let $X1, X2, X3$ be real vectors and $F \geq 0$ a scalar. The differential mutation operator produces a new vector U as follows:

$$U = X3 + F \cdot (X1 - X2) \tag{1}$$

The algebraic operations on real vectors in Equation 1 can be represented graphically [16] as in Figure 1(a).

Unfortunately, this graphical interpretation of Equation 1 in terms of operations on vectors cannot be used to generalize Equation 1 to general metric spaces because algebraic operations on vectors are not well-defined at this level of generality. However, Equation 1 can be rewritten in terms of only convex combinations of vectors. This allows us to interpret graphically this equation in terms of segments and extension rays, which are geometric elements well-defined

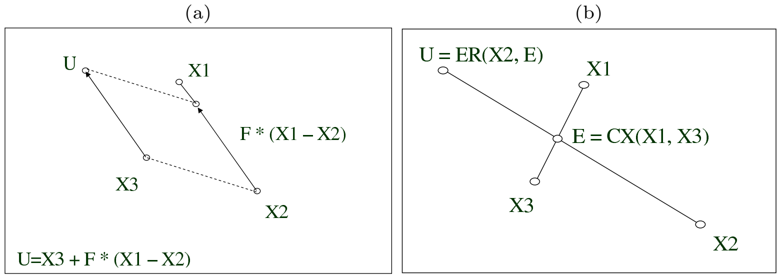

Fig. 1. Construction of U using vectors (a) and construction of U using convex combination and extension ray (b)

on any metric space. Figure 1(b) illustrates the construction of U using convex combination and extension ray. The point E is the intersection point of the segments $[U, X2]$ and $[X1, X3]$. All the distances from E to the endpoints of these segments can be determined from the coefficients of Equation 1 [12]. The point U can therefore be determined geometrically in two steps: (i) determining E as convex combination of $X1$ and $X3$; (ii) projecting $X2$ beyond E (extension ray) obtaining a point U at the known required distance from E. In the Euclidean space, the constructions of U using vectors (Figure 1(a)) and convex combinations (Figure 1(b)) are equivalent. For a detailed description of the relationship between the two interpretations see [12].

Segments and extension rays in the Euclidean space can be expressed in terms of distances, hence, these geometric objects can be naturally generalized to generic metric spaces by replacing the Euclidean distance with a generic metric [12]. The differential mutation operator $U = DM(X1, X2, X3)$ with scale factor F can now be defined for any metric space following the construction of U presented in Figure 1(b) as follows:

1. Compute $W = \frac{1}{1+F}$
2. Get E as the convex combination $CX(X1, X3)$ with weights $(1 - W, W)$
3. Get U as the extension ray $ER(X2, E)$ with weights $(W, 1 - W)$

The weight pair of CX can be thought of indicating the intensity of "linear attraction" of E to $X1$ and $X3$ respectively. So, the larger the weight of $X1$ the closer E will be to it. The weight pair of ER has an analogous meaning where the weights refer to attraction of E to $X2$ and U respectively. However, notice that the unknown in the ER case is a point of attraction (U), rather than the point on which the attraction is exerted (E) as it was the case in CX.

After applying differential mutation, the DE algorithm applies discrete recombination to U and $X(i)$ with probability parameter Cr generating V. This operator can be thought as a weighted geometric crossover and readily generalized as follows: $V = CX(U, X(i))$ with weights $(Cr, 1 - Cr)$ [12].

2.3 Definition of convex combination and extension ray

The notion of convex combination in metric spaces was introduced in the GPSO framework [7]. The convex combination $C = CX((A, W_A), (B, W_B))$ of two

Algorithm 2 Formal Geometric Differential Evolution

```
1: initialize population of $N_p$ configurations at random
2: while stop criterion not met do
3:   for all configuration $X(i)$ in the population do
4:     pick at random 3 distinct configurations from the current population $X1, X2, X3$
5:     set $W = \frac{1}{1+F}$ where $F$ is the scale factor parameter
6:     create intermediate configuration $E$ as the convex combination $CX(X1, X3)$ with weights $(1 - W, W)$
7:     create mutant configuration $U$ as the extension ray $ER(X2, E)$ with weights $(W, 1 - W)$
8:     create candidate configuration $V$ as the convex combination $CX(U, X(i))$ with weights $(Cr, 1 - Cr)$ where $Cr$ is the recombination parameter
9:     if $f(V) \geq f(X(i))$ then
10:      set the $i^{th}$ configuration in the next population $Y(i) = V$
11:    else
12:      set $Y(i) = X(i)$
13:    end if
14:  end for
15:  for all configuration $X(i)$ in the population do
16:    set $X(i) = Y(i)$
17:  end for
18: end while
```

points A and B with weights W_A and W_B (positive and summing up to one) in a metric space endowed with distance function d returns the set of points C such that $d(A, C)/d(A, B) = W_B$ and $d(B, C)/d(A, B) = W_A$ (the weights of the points A and B are inversely proportional to their distances to C). When specified to Euclidean spaces, this notion of convex combination coincides with the traditional notion of convex combination of real vectors.

The notion of extension ray in metric spaces was introduced in the GDE framework [12]. The weighted extension ray ER is defined as the inverse operation of the weighted convex combination CX, as follows. The weighted extension ray $ER((A, w_{ab}), (B, w_{bc}))$ of the points A (origin) and B (through) and weights w_{ab} and w_{bc} returns those points C such that their convex combination with A with weights w_{bc} and w_{ab}, $CX((A, w_{ab}), (C, w_{bc}))$, returns the point B.

The set of points returned by the weighted extension ray ER can be characterized in terms of distances to the input points of ER, as follows [12]. This characterization may be useful to construct procedures to implement the weighted extension ray for specific spaces.

Lemma 1. *The points C returned by the weighted extension ray $ER((A, w_{ab}), (B, w_{bc}))$ are exactly those points which are at a distance $d(A, B) \cdot w_{ab}/w_{bc}$ from B* and *at a distance $d(A, B)/w_{bc}$ from A (see [12] for the proof).*

3 GP-specific search operators for GDE

In order to specify the GDE algorithm to the specific space of genetic programs, we need to choose a distance between genetic programs. A natural choice of distance would be a distance (metric) associated to the Koza-style crossover [4]. This would allow us to derive the specific GDE that searches the same fitness landscape seen by this crossover operator. Unfortunately, the Koza-style crossover is provably non-geometric under any metric [10], so there is no distance associated with it[4] we can use as basis for the GDE. Another crossover operator, the homologous crossover [5] is provably geometric under Structural Hamming Distance (SHD) [9] which is a variant of the well-known structural distance for genetic programming trees [1]. We use this distance as basis for the GDE because we will be able to use the homologous crossover as a term of reference. Notice, however, that in principle, we could choose any distance between genetic programming trees as a basis of the GDE.

3.1 Homologous crossover and Structural Hamming Distance

The common region is the largest rooted region where two parent trees have the same topology. In homologous crossover [5] parent trees are aligned at the root and recombined using a crossover mask over the common region. If a node belongs to the boundary of the common region and is a function then the entire subtree rooted in that node is swapped with it.

The structural distance [1] is an edit distance specific to genetic programming trees. In this distance, two trees are brought to the same tree structure by adding null nodes to each tree. The cost of changing one node into another can be specified for each pair of nodes or for classes of nodes. Differences near the root have more weight. The Structural Hamming Distance [9] is a variant of the structural distance in which, when two matched subtrees have roots of different arities, they are considered to be at a maximal distance (set to 1). Otherwise, their distance is computed as in the original structural distance.

Definition 1. *(Structural Hamming Distance (SHD)). Let T_1 and T_2 be trees, and p and q their roots. Let $hd(p,q)$ be the Hamming distance between p and q (0 if $p = q$, 1 otherwise). Let s_i and t_i be the i^{th} of the m subtrees of p and q.*
$dist(T_1, T_2) = hd(p, q)$ *if* $arity(p) = arity(q) = 0$
$dist(T_1, T_2) = 1$ *if* $arity(p) \neq arity(q)$
$dist(T_1, T_2) = \frac{1}{m+1}(hd(p,q) + \sum_{i=1..m} dist(s_i, t_i))$ *if* $arity(p) = arity(q) = m$

Theorem 1. *Homologous crossover is a geometric crossover under SHD [9].*

3.2 Convex combination

In the following, we first define a weighted version of the homologous crossover. Then we show that this operator is a convex combination in the space of genetic programming trees endowed with SHD. In other words, the weighted homologous crossover implements a convex combination CX in this space.

Definition 2. *(Weighted homologous crossover). Let P_1 and P_2 be two parent trees, and W_1 and W_2 their weights, respectively. Their offspring O is generated using a crossover mask on the common region of P_1 and P_2 such that for each position of the common region, P_1 nodes appear in the crossover mask with probability W_1, and P_2 nodes appear with probability W_2.*

[4] In the sense that there is no distance such that the offspring trees are always within the metric segment between parent trees.

Theorem 2. *The weighted homologous crossover is (in expectation) a convex combination in the space of genetic programming trees endowed with SHD.*

Proof. The weighted homologous crossover is a special case of homologous crossover so it is also geometric under SHD. Therefore, the offspring of the weighted homologous crossover are in the segment between parents as required to be a convex combination. To complete the proof we need to show that the weights W_1 and W_2 of the weighted homologous crossover are inversely proportional to the expected distances $E[SHD(P_1, O)]$, $E[SHD(P_2, O)]$ from the parents P_1 and P_2 to their offspring O, as follows.

Given two trees P_1 and P_2, the SHD can be seen as a weighted Hamming distance on the common region of P_1 and P_2 where the weight w_i on the distance of the contribution of a position i in the common region depends on the arities of the nodes on the path from i to the root node. For each position i of the common region, the expected contribution $SHD_i(P_1, O)$ to the distance $SHD(P_1, O)$ of that specific position is directly proportional to w_i and inversely proportional to the weight W_1 (so, $E[SHD_i(P_1, O)] = w_i/W_1$). This is because, from the definition of weighted homologous crossover, W_1 is the probability that at that position the offspring O equals the parent P_1. So, the higher this probability, the smaller the expected contribution to the distance at that position. Furthermore the contribution to the distance is proportional to the weight w_i of the position i by definition of weighted Hamming distance. From the linearity of the expectation operator, we have that $E[SHD(P_1, O)] = E[\sum_i SHD_i(P_1, O)] = \sum_i E[SHD_i(P_1, O)] = \sum_i w_i/W_1 = 1/W_1$. The last passage holds true because by definition of SHD the sum of the weights on the common region equals 1 (this corresponds to the case of having two trees maximally different on the common region and their distance is 1). Analogously, for the other parent one obtains $E[SHD(P_2, O)] = 1/W_2$. This completes the proof.

3.3 Extension ray

In the following, we first define two weighted homologous recombinations. Then we show that these operators are extension ray recombinations in the space of genetic programming trees endowed with SHD. The first recombination produces offspring with the same tree structure as the second parent. The second recombination is more general and can produce offspring with tree structure different from both parents. From a geometric viewpoint, these weighted homologous recombinations implement two different versions of extension ray recombination ER in the space of genetic programming trees endowed with SHD, where the first operator produces a subset of the points produced by the second operator.

To determine a recombination that implements an extension ray operator, it is useful to think of an extension ray operator, algebraically, as the inverse operation of a convex combination operator. In the convex combination, the unknown is the offspring C that can be determined by combining the known parents P_1 and P_2. In the extension ray, the distance relationship between P_1, P_2 and C is the same as in the convex combination, but P_1 (the origin of the extension ray) and C (the point the extension ray passes through) are known, and P_2 (the point on the extension ray) is unknown, i.e., $C = CX(P_1, P_2)$ can be equivalently rewritten as $P_2 = ER(P_1, C)$ depending whether C or P_2 is the unknown.

The first weighted extension ray homologous recombination is described in Algorithm 3. The second recombination is the same operator as the first with the following addition before line 6 in Algorithm 3. In the common region, if two subtrees $S_A(i)$ and $S_B(i)$ coincide in structure and contents (not only if their root nodes $T_A(i)$ and $T_B(i)$ coincide), put in the corresponding position i in the offspring T_C a random subtree S_C (with in general different structure and contents from S_A and S_B). Skip the remaining nodes in the common region covered by $S_A(i)$ and $S_B(i)$.

Notice that, in the definition of the second recombination above, any arbitrarily large subtree S_C could be generated to be included in T_C. However, in the implementation, its size should be limited. In the experiment, we generate S_C with the same number of nodes as S_A and S_B.

Algorithm 3 Weighted extension ray homologous recombination 1

```
    Inputs: parent trees $T_A$ (origin point of the ray) and $T_B$ (passing through point of the ray), with
    corresponding weights $w_{AB}$ and $w_{BC}$ (both weights are between 0 and 1 and sum up to 1)
    Output: a single offspring tree $T_C$ (a point on the extension ray beyond $T_B$ on the ray originating
    in $T_A$ and passing through $T_B$)
 1: compute the Structural Hamming Distance $SHD(T_A, T_B)$ between $T_A$ and $T_B$
 2: set $SHD(T_B, T_C) = SHD(T_A, T_B) \cdot w_{AB}/w_{BC}$ (compute the distance between $T_B$ and $T_C$ using
    the weights)
 3: set $p = SHD(T_B, T_C)/(1 - SHD(T_A, T_B))$ (the probability $p$ of flipping nodes in the common
    region away from $T_A$ and $T_B$ beyond $T_B$)
 4: set $T_C = T_B$
 5: for all position $i$ in the common region between $T_A$ and $T_B$ do
 6:     consider the paired nodes $T_B(i)$ and $T_A(i)$ in the common region
 7:     if $T_B(i) = T_A(i)$ and $p >$ random number between 0 and 1 then
 8:         set $T_C(i)$ to a random node with the same arity of $T_A(i)$ and $T_B(i)$
 9:     end if
10: end for
11: return tree $T_C$ as offspring
```

Theorem 3. *The weighted extension homologous ray recombinations 1 and 2 are (in expectation) extension ray operators in the space of genetic programming trees endowed with SHD.*

Proof. First we prove that $T_C = ER(T_A, T_B)$ by showing that $T_B = CX(T_A, T_C)$. Then we prove that the expected distances $E[SHD(T_A, T_B)]$ and $E[SHD(T_B, T_C)]$ are inversely proportional to the weights w_{AB} and w_{BC}, respectively.

Let us consider recombination 1. The offspring T_C has the same structure of T_B. This is because T_C was constructed starting from T_B and then for each node of the common region between T_A and T_B, T_C was not changed or it was randomly chosen but preserving the arity of that node in T_B.

The structures of the common regions $CR(T_A, T_B)$ and $CR(T_A, T_C)$ coincide. This is because the structure of the common region between two trees is only function of their structures. So, since T_B and T_C have the same structure, $CR(T_A, T_B)$ and $CR(T_A, T_C)$ have the same structure.

The tree T_B can be obtained by homologous crossover applied to T_A and T_C (hence, $T_C = ER(T_A, T_B)$). This can be shown considering two separate cases, (i) nodes of T_B

inherited from the common region $CR(T_A, T_C)$ and (ii) subtrees of T_B inherited from subtrees of T_A and T_C at the bottom of the common region. Let us consider nodes on the common region. For each node with index i in the common region, the node $T_B(i)$ matches $T_A(i)$ or $T_C(i)$. This is true from the way $T_C(i)$ was chosen on the basis of the values of $T_A(i)$ and $T_B(i)$. We have two cases. First, $T_C(i)$ was chosen at random, when $T_A(i) = T_B(i)$. In this case $T_B(i)$ can be inherited from $T_A(i)$, since it may be $T_B(i) \neq T_C(i)$ but $T_B(i) = T_A(i)$. Second, $T_C(i)$ was chosen to equal $T_B(i)$, when $T_A(i) \neq T_B(i)$. In this case $T_B(i)$ can be inherited from $T_C(i)$. In either cases, for nodes on the common region the corresponding nodes of T_B can be inherited from T_A or T_C. The subtrees of T_B at the bottom of the common region can be inherited all from T_C (both structures and contents). Since by construction T_C inherited those subtrees from T_B without modifying them.

To show that recombination 1 is a weighted extension homologous ray recombination, we are left to show that the expected distances $E[SHD(T_A, T_B)]$ and $E[SHD(T_B, T_C)]$ are inversely proportional to the weights w_{AB} and w_{BC}. The probability p of flipping nodes in the common region away from T_A and T_B beyond T_B was chosen as an appropriate function of w_{AB} and w_{BC} and of $SHD(T_A, T_B)$ to obtain $SHD(T_B, T_C)$ such that the above requirement holds true. It is possible to prove that the chosen p is the correct one using the same argument used in the proof of theorem 2.

Let us consider now recombination 2. In this case, the offspring T_C by construction may have structure different from T_A and T_B. Also, the structures of the common regions $CR(T_A, T_B)$ and $CR(T_A, T_C)$ do not coincide. The structure of $CR(T_A, T_C)$ is covered by the structure of $CR(T_A, T_B)$ ($CR(T_A, T_C)$ is a substructure of $CR(T_A, T_B)$). The part of $CR(T_A, T_B)$ that does not cover $CR(T_A, T_C)$ comprises subtrees that are identical in structures and contents in T_A and T_B.

The tree T_B can be obtained by homologous crossover applied to T_A and T_C (hence, $T_C = ER(T_A, T_B)$). This can be shown similarly as for recombination 1 but with an extra case to consider. Nodes of T_B corresponding to nodes in the common region $CR(T_A, T_C)$ can be inherited from T_A or T_B. The subtrees of T_B at the bottom of the common region $CR(T_A, T_C)$ can be inherited all from T_C (both structures and contents). The extra case is for the subtrees of T_B that are in the part of $CR(T_A, T_B)$ that does not cover $CR(T_A, T_C)$. These subtrees cannot be inherited from T_C, which differs form T_B by construction, but they can always be inherited from T_A.

As for the requirement on the expected distances being inversely proportional to the weights, the probability p can be chosen as the case for recombination 1 due to the recursive definition of SHD that treats nodes and subtrees uniformly.

Now we have operational definitions of convex combination and extension ray for the space of genetic programming trees under SHD. These space-specific operators can be plugged in the formal GDE (Algorithm 2) to obtain a specific GDE for the genetic programming trees space, the GDE-GP.

4 Experiments

This section reports an initial experimental analysis of the GDE-GP behavior on four standard GP benchmark problems: Symbolic Regression of the quartic polynomial, Artificial Ant on the Santa Fe trail, 5-Bit Even Parity, and 11-Bit Multiplexer. In all these problems fitness is calculated so that lower values are better. All the experiments used $F = 0.8$ and $Cr = 0.9$, according to [16]. Both extension ray recombinations 1 and 2 were tested, giving rise to distinct techniques we designate as GDE1 and GDE2. As a baseline for comparison we used standard GP with homologous crossover (70%) and reproduction (30%), always applying point mutation with probability $1/L$, where L is the number of nodes of the individual. We call this baseline HGP. All the experiments were performed using populations of two different sizes (500 and 1000 individuals) initialized with the Ramped Half-and-Half procedure [4] with an initial maximum depth of 8, allowed to evolve for 50 generations. Each experiment was repeated 20 times. Statistical significance of the null hypothesis of no difference was determined with pairwise Kruskal-Wallis non-parametric ANOVAs at $p = 0.05$.

Figure 2 shows the boxplots of the best fitness achieved along the run, using populations of 500 individuals (top row) and 1000 individuals (bottom row). With a population size of 500, in all four problems there is a statistically significant difference between HGP and each of the GDE-GP techniques, and no significant difference between GDE1 and GDE2. GDE-GP is consistently better than HGP, regardless of the extension ray recombination used.

It may be argued that HGP is being crippled by such a small population size, which may reduce diversity along the run. This could be true, because when doubling the population size HGP significantly improves its best fitness of run in all except the Parity problem. However, the GDE-GP techniques also show significant improvements in most cases, and remain consistently better than HGP, regardless of the extension ray recombination used, exactly as before.

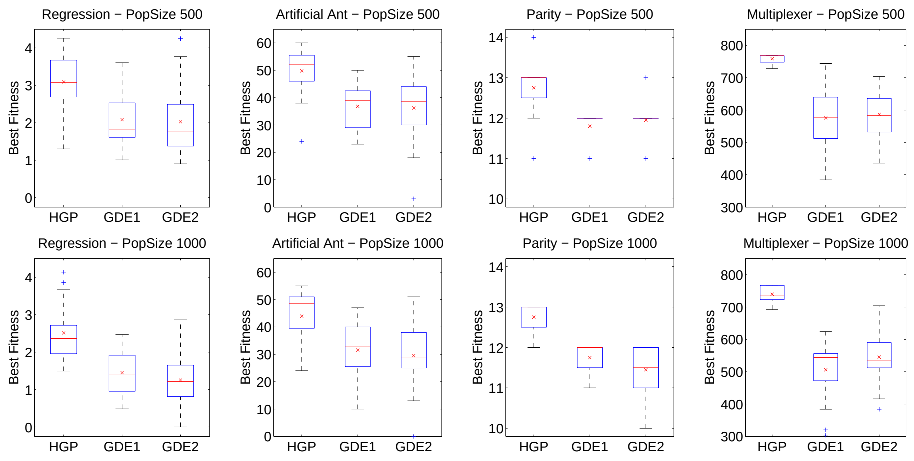

Fig. 2. Boxplots of the best fitness achieved in each problem (× marks the mean). Population sizes of 500 individuals (top row) and 1000 individuals (bottom row)

However, the observation of diversity, measured as the percentage of genotypically distinct individuals in the population, revealed somewhat unexpected results. Figure 3 (top row) shows the evolution of the median values of diversity along the run, for both population sizes. Not only HGP shows no clear signs of

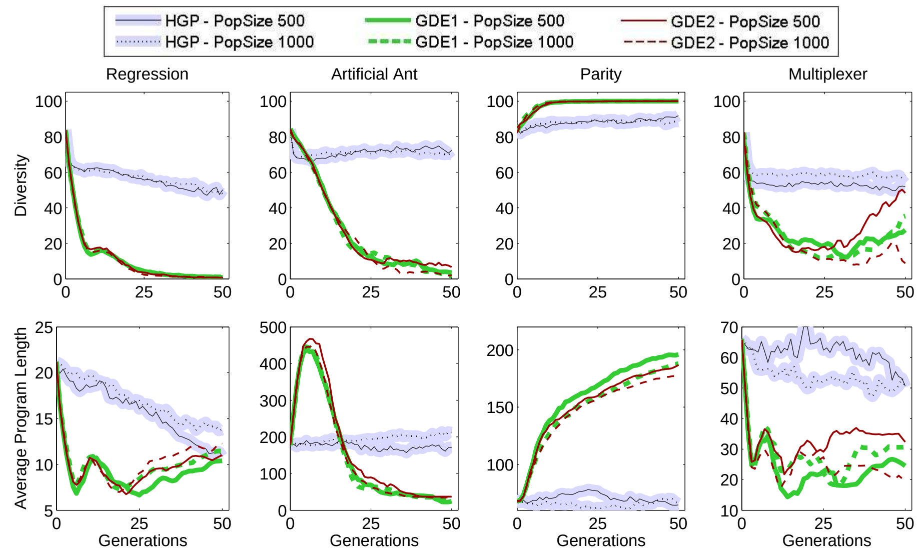

Fig. 3. Evolution of the median values of diversity (top row) and average program length (bottom row) in each problem

diversity loss, regardless of population size, but GDE-GP exhibits an extraordinarily varied behavior, approaching both extreme values in different problems (in Regression and Artificial Ant it practically reaches 0% while in Parity it reaches 100%), in some cases undergoing large fluctuations along the run (Multiplexer).

Finally, in Figure 3 (bottom row) we look at the evolution of the median values of average program length along the run, for both population sizes. Once again GDE-GP behaves radically differently from HGP, with both GDE1 and GDE2 presenting large but smooth fluctuations in most problems, when compared to the more constrained but somewhat erratic behavior of HGP. The most interesting case is probably the Artificial Ant, where GDE-GP quickly and steadily increases the average program length until a plateau is reached, followed by a steep decrease to very low values. Curiously, there is no correspondingly interesting behavior in terms of the evolution of fitness (not shown), at least when observed in median terms. Only in the Parity problem GDE-GP exhibits a behavior that would be expected in standard (with subtree crossover) GP runs.

5 Conclusions

Geometric DE is a generalization of the classical DE to general metric spaces. In particular, it applies to combinatorial spaces. In this paper we have demonstrated how to specify the general Geometric Differential Evolution algorithm to the space of genetic programs. We have reported interesting experimental results where the new algorithm performs better than regular GP with homologous crossover in four typical GP benchmarks using different population sizes. In terms of diversity and average program length, neither technique seems to be largely influenced by the population size, most differences being the product of large individual variations. In the future we will deepen our study of the interesting dynamics revealed by the new algorithm.

Acknowledgments. We would like to thank Riccardo Poli for passing us the code of the homologous crossover for genetic programs. The second author also acknowledges project PTDC/EIACCO/103363/2008 from Fundação para a Ciência e a Tecnologia, Portugal.

References

1. A. Ekart and S. Z. Nemeth, *A Metric for Genetic Programs and Fitness Sharing*, In Proceedings of EuroGP-2000, Springer, 2000, pp. 259–270.
2. T. Gong and A. L. Tuson, *Differential Evolution for Binary Encoding*, Soft Computing in Industrial Applications, Springer, 2007, pp. 251–262.
3. J. Kennedy and R. C. Eberhart, *Swarm Intelligence*, Morgan Kaufmann, 2001.
4. J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, The MIT Press, 1992.
5. W. Langdon and R. Poli, *Foundations of Genetic Programming*, Springer, 2002.
6. A. Moraglio, *Towards a Geometric Unification of Evolutionary Algorithms*, Ph.D. thesis, University of Essex, 2007.
7. A. Moraglio, C. Di Chio, and R. Poli, *Geometric Particle Swarm Optimization*, In Proceedings of EuroGP-2007, Springer, 2007, pp. 125–136.
8. A. Moraglio, C. Di Chio, J. Togelius, and R. Poli, *Geometric Particle Swarm Optimization*, Journal of Artificial Evolution and Applications, 2008, Article ID 143624.
9. A. Moraglio and R. Poli, *Geometric Landscape of Homologous Crossover for Syntactic Trees*, Proceedings of CEC-2005, IEEE Press, 2005, pp. 427–434.
10. A. Moraglio and R. Poli, *Inbreeding Properties of Geometric Crossover and Non-geometric Recombinations*, In Proceedings of the Workshop on the Foundations of Genetic Algorithms, Springer, 2007, pp. 1–14.
11. A. Moraglio and J. Togelius, *Geometric PSO for the Sudoku Puzzle*, In Proceedings of GECCO-2007, ACM Press 2007, pp. 118–125.
12. A. Moraglio and J. Togelius, *Geometric Differential Evolution*, In Proceedings of GECCO-2009, ACM Press, 2009, pp. 1705–1712.
13. M. O'Neill and A. Brabazon, *Grammatical Differential Evolution*, In Proceedings of ICAI-2006, CSREA Press, 2006, pp. 231–236.
14. G. C. Onwubolu and D. Davendra (eds.), *Differential Evolution: A Handbook for Global Permutation-based Combinatorial Optimization*, Springer, 2009.
15. G. Pampara, A.P. Engelbrecht, and N. Franken, *Binary Differential Evolution*, In Proceedings of CEC-2006, IEEE Press, 2006, pp. 1873–1879.
16. K. V. Price, R. M. Storm, and J. A. Lampinen, *Differential Evolution: A Practical Approach to Global Optimization*, Springer, 2005.
17. J. Togelius, R. De Nardi, and A .Moraglio, *Geometric PSO + GP = Particle Swarm Programming*, In Proceedings of CEC-2008, IEEE Press, 2008, pp. 3594–3600.