Geometric Nelder-Mead Algorithm on the Space of Genetic Programs

Alberto Moraglio
School of Computing and Centre for Reasoning
University of Kent
Canterbury, UK
a.moraglio@kent.ac.uk

Sara Silva
INESC-ID Lisboa and Center for Informatics and Systems of the University of Coimbra
Portugal
sara@{kdbio.inesc-id.pt,dei.uc.pt}

ABSTRACT

The Nelder-Mead Algorithm (NMA) is a close relative of Particle Swarm Optimization (PSO) and Differential Evolution (DE). In recent work, PSO, DE and NMA have been generalized using a formal geometric framework that treats solution representations in a uniform way. These formal algorithms can be used as templates to derive rigorously specific PSO, DE and NMA for both continuous and combinatorial spaces retaining the same geometric interpretation of the search dynamics of the original algorithms across representations. In previous work, a geometric NMA has been derived for the binary string representation and permutation representation. Furthermore, PSO and DE have already been derived for the space of genetic programs. In this paper, we continue this line of research and derive formally a specific NMA for the space of genetic programs. The result is a Nelder-Mead Algorithm searching the space of genetic programs by acting directly on their tree representation. We present initial experimental results for the new algorithm. The challenge tackled in the present work compared with earlier work is that the pair NMA and genetic programs is the most complex considered so far. This combination raises a number of issues and casts light on how algorithmic features can interact with representation features to give rise to a highly peculiar search behaviour.

Categories and Subject Descriptors

F.2 [**Theory of Computation**]: Analysis of Algorithms and Problem Complexity

General Terms

Theory

Keywords

Genetic Programming, Nelder-Mead Algorithm, Search Operator Design

GECCO'11, July 12–16, 2011, Dublin, Ireland.

1. INTRODUCTION

The Nelder-Mead Algorithm [16] is a numerical optimization method widely used in practice. Contrasted with the majority of classic methods for numerical optimization, it only uses the values of the objective function without any derivative information. The search done by NMA is based on geometric operations (reflection, expansion, contraction and shrinking) on a current set of points, seen as the corners of a n-dimensional polygon (a simplex), to determine what points in space to evaluate next. The overall behaviour of the NMA expands or focuses the search adaptively on the basis of the topography of the fitness landscape.

Interestingly, the NMA can be seen as a form of (population-based) evolutionary algorithm with special selection and reproduction operators [20]. Also, there are similarities between the search operators employed by the NMA and those of DE and PSO that have led a number of authors to propose hybrid approaches (see for example [22] and [6]). As the original versions of DE and PSO, NMA requires the search space to be continuous and the points in space to be represented as vectors of real numbers.

There are few extensions of DE and PSO to combinatorial spaces [19] [18] [3] [1] and to the space of genetic programs [17]. 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 search algorithms to solve these continuous problems. Other works present PSO and DE algorithms defined on combinatorial spaces acting directly on the original solution representation that, however, are only loosely related to the traditional algorithms 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 search algorithm needs to be rethought and adapted to the new representation. Apart from very recent work [10], there are no generalizations of the NMA to combinatorial spaces.

The searches done by PSO, DE and NMA have natural geometric interpretations and can be understood as the motion of points in space obtained by different but related linear combinations of their current and past positions to determine their new positions. Geometric Particle Swarm Optimization (GPSO) [8], Geometric Differential Evolution (GDE) [15] and Geometric Nelder-Mead Algorithm (GNMA) [10] are recently devised formal generalizations of PSO, DE and NMA that, in principle, can be specified to any solution representation while retaining the original geometric interpretation of the dynamics of the points in space across rep-

resentations. In particular, these formal algorithms can be applied to any search space endowed with a distance and associated with any solution representation to derive formally specific PSO, DE and NMA for the target space and for the target representation.

Specific GPSOs were derived for different types of continuous spaces and for the Hamming space associated with binary strings [9], for spaces associated with permutations [14] and for spaces associated with genetic programs [21]. GDE was specialized to the space of binary strings [15] and, very recently, to the space of genetic programs [13]. GNMA was specialized to the space of binary strings and to spaces associated with permutations [10]. The derived algorithms performed satisfactorily in experimental results. This suggests that the generalization methodology employed is a promising one. In this paper, we continue this line of research and derive the Geometric Nelder-Mead Algorithm for spaces associated with genetic programs. Preliminary experimental results indicate that the GP-based GNMA does not perform as well as the GNMA for other spaces. The reason behind it seems to be related with the peculiar geometric properties of the space of genetic programs. We present an analysis aimed at understanding what features of the GP tree representation makes it less suitable to be searched with GNMA. This casts light on how some algorithmic features can interact with some representation features to give rise to a highly peculiar search behaviour. This knowledge may be helpful to discriminate between desirable and less desirable features of the combination algorithm-representation to obtain a successful outcome when considering applying the geometric framework to generalise algorithms from continuous to combinatorial spaces.

2. CLASSIC NELDER-MEAD ALGORITHM

In this section, we describe the traditional NMA [16] (see Algorithm 1). The NMA uses $n+1$ points in $\mathbf{R}^n$. These points form a type of n-dimensional polygon, a simplex, which has $n+1$ points as vertices in $\mathbf{R}^n$. For example, the simplex is a triangle in $\mathbf{R}^2$ and a tetrahedron in $\mathbf{R}^3$. The initial simplex has to be non-degenerate, i.e., the points must not lie in the same hyperplane. This allows the NMA to search in all n dimensions. The method then performs a sequence of transformations of the simplex, which preserve non-degeneracy, aimed at decreasing the function values at its vertices. At each step, the transformation is determined by computing one or more test points and comparing their function values. In Figure 1, we illustrate the NMA transformations for the two-dimensional case, where the simplex S consists of three points.

The optimization process described by Algorithm 1 starts with creating a sample of $n+1$ random points in the search space. Notice that, apart from the creation of the initial simplex, all further steps are deterministic and do not involve random choices. In each loop iteration, the points in the simplex S are arranged in ascending order according to their corresponding objective values. Hence, the best solution candidate is $S[0]$ and the worst is $S[n]$. We then compute the center M of the n best points and then reflect the worst candidate solution $S[n]$ through this point, obtaining the new point R as also illustrated in Fig. 1(a). The reflection parameter α is usually set to 1. In the case that R is neither better than $S[0]$ nor as worse as $S[n]$, we directly replace $S[n]$ with it. If R is better than the best

Algorithm 1 Nelder-Mead Algorithm

```
Input: f: the objective function to minimize
Input: n+1: number of points in the simplex
Input: α, ρ, γ, σ: reflection, expansion, contraction and shrink coefficients
Output: x*: the best solution found

S ← createPop(n+1)
while stop criterion not met do
   S ← sortPop(S, f)
   // Center of mass: determine the center of mass of the n best points
   M ← (1/n) Σ_{i=0,n-1} S[i]
   // Reflection: reflect the worst point over M
   R ← M + α(M − S[n])
   if f(S[0]) < f(R) < f(S[n]) then
      S[n] ← R
   else
      if f(R) ≤ f(S[0]) then
         // Expansion: try to search farther in this direction
         E ← R + γ(R − M)
         if f(E) < f(R) then
            S[n] ← E
         else
            S[n] ← R
         end if
      else
         b ← true
         if f(R) ≥ f(S[n − 1]) then
            // Contraction: a test point between R and M
            C ← ρR + (1 − ρ)M
            if f(C) < f(R) then
               S[n] ← C
               b ← false
            end if
         end if
         if b = true then
            // Shrink towards the best solution candidate S[0]
            for i from n down to 1 do
               S[i] ← S[0] + σ(S[i] − S[0])
            end for
         end if
      end if
   end if
end while
return S[0]
```

solution candidate $S[0]$, we expand the simplex further into this promising direction. As sketched in Fig. 1(b), we obtain the point E with the expansion parameter γ set to 1. We now take the best of these two points to replace $S[n]$. If R is no better than $S[n]$, the simplex is contracted by creating a point C somewhere in between R and M. In Fig. 1(c), the contraction parameter ρ was set to 1/2. We substitute $S[n]$ with C only if C is better than R. When everything else fails, we shrink the whole simplex by moving all points (except $S[0]$) into the direction of the current optimum $S[0]$. The shrinking parameter σ normally has the value 1/2, as is the case in the example outlined in Fig. 1(d).

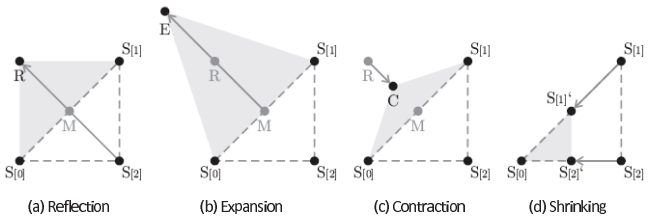

Figure 1: One step of the NMA in R^2 (figure modified from [23])

3. GEOMETRIC NMA

In this section, we present how the general Geometric Nelder-Mead Algorithm [10] (Algorithm 2) was derived from the classic Nelder-Mead Algorithm (Algorithm 1). The generalization was obtained using a methodology to generalize search algorithms for continuous spaces to combinatorial spaces [15] based on the geometric framework introduced by Moraglio [7], sketched in the following.

1. 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.

2. 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.

3. 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.

4. 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, DE and NMA to any metric space, obtaining GPSO, GDE and GNMA, then to derive the specific search operators for a number of specific representations and distances. In particular for GNMA, the generalization of the classic Nelder-Mead Algorithm to general metric spaces was done by recasting the search operations described in the previous section (reflection, expansion, contraction and shrinking) as functions of the distance of the underlying search space, thereby obtaining their abstract geometric definitions, as explained below. Then, the specific GNMA for the Hamming space associated with binary strings was derived. Analogously, the specific GNMA for the space of permutations with swap distance was derived by plugging this distance in the abstract definition of the search operators. In Section 4, we will derive the specific GNMA for the space of genetic programs by using the abstract definition with a distance function between GP trees.

3.1 Geometric Generalization of the Nelder-Mead Algorithm

Using the notion of convex combination CX, extension ray ER and center of mass CM we can generalize all search operators of the classical Nelder-Mead Algorithm from the Euclidean case to generic metric spaces because, as we will see in the following section, these are geometric elements well-defined on any metric space.

The graphical description of the search operations of NMA (Fig. 1) leads directly to their geometric interpretation in terms of convex combination and extension ray, as follows. The reflection of the worst point $S[n]$ over M can be seen as picking a point beyond M on the extension ray originating in $S[n]$ and passing through M. The expansion operation can be seen as picking a point beyond R on the extension ray originating in M and passing through R. The contraction operation can be seen as picking a point in the segment between R and M. The shrink of all points $S[i]$ towards the best in the population $S[0]$ can be seen as replacing each point $S[i]$ with a point in the segment between $S[i]$ and $S[0]$.

Algorithm 2 Formal Nelder-Mead Algorithm

```
1: Input: f: the objective function to minimize
2: Input: n + 1: number of points in the simplex
3: Input: α, ρ, γ, σ: reflection, expansion, contraction and shrink
   coefficients
4: Output: x*: the best solution candidate found
5:
6: S ← createPop(n + 1)
7: while stop criterion not met do
8:    S ← sortPop(S, f)
9:    // Center of mass: determine the center of mass of the n best
      points
10:   M ← CM(S[0], S[1], ..., S[n − 1])
11:   // Reflection: reflect the worst point over m
12:   R ← ER(S[n], M) with weights (α/(1+α), 1/(1+α))
13:   if f(S[0]) < f(R) < f(S[n]) then
14:      S[n] ← R
15:   else
16:      if f(R) ≤ f(S[0]) then
17:         // Expansion: try to search farther in this direction
18:         E ← ER(M, R) with weights (1/γ, (γ−1)/γ)
19:         if f(E) < f(R) then
20:            S[n] ← E
21:         else
22:            S[n] ← R
23:         end if
24:      else
25:         b ← true
26:         if f(R) ≥ f(S[n − 1]) then
27:            // Contraction: a test point between R and M
28:            C ← CX(R, M) with weights (ρ, 1 − ρ)
29:            if f(C) < f(R) then
30:               S[n] ← C
31:               b ← false
32:            end if
33:         end if
34:         if b = true then
35:            // Shrink towards the best solution candidate S[0]
36:            for i from n down to 1 do
37:               S[i] ← CX(S[0], S[i]) with weights (1 − σ, σ)
38:            end for
39:         end if
40:      end if
41:   end if
42: end while
43: return S[0]
```

In the following, we rewrite the algebraic definitions of the search operations of NMA to determine the weights of the corresponding convex combination or extension ray combination.

The definition of the reflection operation is $R = M + \alpha(M - S[n])$ (see Algorithm 1, line 12) and it can be rewritten as $M = \frac{\alpha}{1+\alpha}S[n] + \frac{1}{1+\alpha}R$. Since the coefficients of $S[n]$ and R are positive and sum up to 1 (for $\alpha \in [0, 1]$), this equation says that M is the convex combination of $S[n]$ and R with those coefficients. However, since R is the unknown and $S[n]$ and M are given, we can determine R as the inverse operation of the convex combination above, which is the extension ray combination with origin in $S[n]$ passing through M and keeping the same weights $(\frac{\alpha}{1+\alpha}, \frac{1}{1+\alpha})$ of the convex combination.

The definition of the expansion operation is $E = R + \gamma(R - M)$ (see Algorithm 1, line 18) and it can be rewritten as $R = \frac{1}{\gamma}M + \frac{\gamma-1}{\gamma}E$, which for $\gamma > 1$ is a convex combination of M and E returning R. Analogously to the reflection operation, since E is unknown and M and R are given, we can

determine E by the extension ray combination with origin in M passing through R with weights $(\frac{1}{\gamma}, \frac{\gamma-1}{\gamma})$.

The definition of the contraction operation is $C = \rho R + (1-\rho)M$ (see Algorithm 1, line 28), which for $\rho \in [0,1]$ is a convex combination of R and M with weights $(\rho, 1-\rho)$ returning C.

The definition of the shrink operation for a point $S[i]$ is $S[i]' = S[0] + \sigma(S[i] - S[0])$ (where $S[i]'$ denotes $S[i]$ at the next time step) (see Algorithm 1, line 37). This can be rewritten as $S[i]' = (1-\sigma)S[0] + \sigma S[i]$, which for $\sigma \in [0,1]$ is a convex combination of $S[0]$ and $S[i]$ with weights $(1-\sigma, \sigma)$ returning $S[i]'$.

By replacing in Algorithm 1 the original operations defined on the Euclidean space with their generalized definitions we obtain the definition of a Formal Nelder-Mead Algorithm valid for any metric space (see Algorithm 2).

3.2 Convex Combination, Extension Ray and Centre of Mass

Center of mass, segments and extension rays in the Euclidean space and their weighted extensions 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.

Let (S, d) be a metric space. A (metric) segment is a set of the form $[x; y] = \{z \in S | d(x,z) + d(z,y) = d(x,y)\}$ where $x, y \in S$. The notion of convex combination in metric spaces was introduced in the GPSO framework [8]. The convex combination $C = CX((A, W_A), (B, W_B))$ of two 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 in the segment $[A; B]$ 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 extension ray $ER(A,B)$ in the Euclidean plane is a semi-line originating in A and passing through B (note that $ER(A,B) \neq ER(B,A)$). The notion of extension ray in metric spaces was introduced in the GDE framework [15]. 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 notion of center of mass was generalized to generic metric spaces in the GNMA framework [10] , as follows. The center of mass CM of a set of points $p_1, ..., p_n$ in a metric space (S, d) is the point $p \in S$ that minimizes its average distance to that set of points, i.e. $CM(p_1, ..., p_n) = argmin_{p \in S} \frac{\sum_{i=1...n} d(p_i, p)}{n}$.

4. GNMA SEARCH OPERATORS FOR GENETIC PROGRAMS

In order to specify the GNMA 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 GNMA that searches the same fitness landscape seen by this crossover operator. Unfortunately, the Koza-style crossover is provably non-geometric under any metric [12], so there is no distance associated with it[1] we can use as basis for the GNMA. Another crossover operator, the homologous crossover [5] is provably geometric under Structural Hamming Distance (SHD) [11] which is a variant of the well-known structural distance for genetic programming trees [2]. We use this distance as basis for the GNMA 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 GNMA. For each distance, we would obtain a different GNMA searching the genetic programs space. In the following, we derive formally specific convex combination, extension ray recombination and center of mass operator for the space of genetic programs under SHD.

4.1 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 [2] 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 [11] 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. *[11] Homologous crossover is a geometric crossover under SHD.*

4.2 Convex combination

The following definition presents a weighted version of the homologous crossover that was introduced in the GDE framework [13]. 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.

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

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.*

THEOREM 2. *[13] The weighted homologous crossover is (in expectation) a convex combination in the space of genetic programming trees endowed with SHD.*

4.3 Extension ray

Algorithm 3 reports a weighted homologous recombination which was originally introduced in the GDE framework [13]. This operator is an extension ray recombination in the space of genetic programming trees endowed with SHD.

Algorithm 3 Weighted extension ray homologous recombination

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)

compute the Structural Hamming Distance $SHD(T_A, T_B)$ between T_A and T_B
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)
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)
set $T_C = T_B$
for all position i in the common region between T_A and T_B **do**
    consider the paired nodes $T_A(i)$ and $T_B(i)$ in the common region and the subtrees $S_A(i)$ and $S_B(i)$ rooted in those nodes
    if $S_B(i) = S_A(i)$ **then**
        (if the subtrees match in structure and contents)
        create a random subtree T_C and root it in the offspring at position i
        set i to skip the remaining nodes in the common region covered by $S_A(i)$ and $S_B(i)$
    end if
    if $T_B(i) = T_A(i)$ and $p >$ random number between 0 and 1 **then**
        set $T_C(i)$ to a random node with the same arity of $T_A(i)$ and $T_B(i)$
    end if
end for
return tree T_C as offspring

THEOREM 3. *[13] The weighted extension homologous ray recombination is (in expectation) an extension ray operator in the space of genetic programming trees endowed with SHD.*

4.4 Centre of Mass

When specified to the Hamming space on binary strings the centre of mass CM coincides with the multi-parental recombination that returns the offspring by taking position-wise the majority vote of the parents [10]. This result generalizes to genetic programming trees under Structural Hamming Distance. Intuitively, the centre of mass tree can be determined via a majority vote for each position in the parent trees which keeps adequately into account their structures by passing the structure (arity) of the most common nodes at that position to the corresponding position in the centre of mass tree. A multi-parental operator following this line of thinking is reported in Algorithm 4. The next theorem shows that this is indeed a center of mass operator.

Algorithm 4 Center of Mass Operator

inputs: parent trees $P_1, P_2, ..., P_n$
output: centre of mass tree p_{cm}
Determine the most common root node of the parent trees.
Let R be this node, let A be its arity, and let S be the set of parent trees whose root node has arity A.
Assign R to the offspring tree p_{cm} at the current position.
if R is terminal **then**
    Return p_{cm}
end if
for all branch i (from 1 to A) **do**
    Determine recursively the center of mass of the subtrees of the branch i of the root nodes of the trees in S
    Assign the center of mass of the branch i to the corresponding branch i of the current node in the offspring tree p_{cm}
end for
Return p_{cm}

THEOREM 4. *The operator in Algorithm 4 is a centre of mass operator in the space of genetic programming trees endowed with SHD.*

PROOF. The centre of mass tree is the tree which is in average closest to all parent trees in SHD, which is the tree which minimises the sum of the distances from it to all parent trees.

From the definition of SHD, the distance is a linear combination of paired nodes contributions with nodes closer to the root node having a larger weight. A pair of matched nodes does not contribute to the distance only when they coincide.

By construction the root node of the centre of mass tree, which is the one with higher weight, gives the minimal contribution to the sum of the SHD to all parent trees because it is chosen to be the node which occurs the most. This choice of node is optimal independently from the subsequent choices of nodes to include in the offspring. Now, this reasoning can be applied recursively to all child subtrees of the root node w.r.t. the parent trees that are left to be compatible with the chosen structure (arity) of the root node. So, an optimal decision in terms of keeping the contribution to the sum of distances minimal is when the roots of these subtrees in the centre of masss tree are chosen to concur with the root nodes that occur the most in the compatible parent trees in the corresponding subtrees. By induction on all levels of the tree, the centre of mass tree so generated is the one at minimum average distance from the parent trees. □

Unlike for the Euclidean case in which the simplex is maintained non-degenerate throughout the search process, so guaranteeing that any dimension is actually being searched, this does not hold true for the cases of the Hamming space and GP spaces. To counteract the degeneracy of the simplex, in the experiments we will use a randomized version of the CM operator which uses the frequency of the most frequent element at each position in the parents as the probability of the offspring to have that element at that position rather than fixing that element deterministically. The expected offspring of the randomized operator is the one obtained with the Algorithm 4, but the variance of the output gives a greater chance to the search of staying open in all dimensions.

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

obtain a specific GNMA for the genetic programming trees space, the GNMA-GP.

5. EXPERIMENTS AND DISCUSSION

This section reports an initial experimental analysis of the GNMA-GP behavior on the classic problem of Symbolic Regression of the quartic polynomial [4] and on a unimodal problem on the space of genetic programs under SHD, in which the fitness of a tree (to minimise) is given by its distance to an arbitrary (i.e., sampled at random) but fixed tree. The last problem can be seen as a generalisation of the OneMax problem for binary strings [2], so we will refer to it as the One-Max-like problem. The reason we included the latter problem in the test-bed is because we wanted to test the GNMA-GP under controlled conditions on a problem whose topographic features of the fitness landscape are explicit and completely understood.

In preliminary experiments, we have found out that the GNMA when specified to the space of Genetic Programs, unlike the case when it is specified to the Hamming space on binary strings, tends to lose population diversity very rapidly. The cause of this seems to be that the average SHD between programs in the initial population is maximal (=1), and that when the GNMA search operators are applied to maximally different trees they become degenerate and return always a clone of a parent as offspring. In particular, operators based on the extension ray recombination become degenerate because when parents are at maximal distance, as there is no space beyond the second parent, the second parent is always returned as offspring. This does not occur in the Hamming space because the average Hamming distance in the initial population is half of the size of the diameter of the space (i.e., of the maximal distance between binary strings), and this allows for plenty of space to generate offspring beyond the second parent.

We attempted to resolve the problem above by recreating a distribution of the distances between individuals in the initial population of Genetic Programs similar to that obtained with binary strings. This can be achieved by sampling at random a single tree at first, and then creating an initial population with trees obtained applying random walks originating in that tree. A random walk of length n is a sequence of n repeated mutations applied to the same individual. The average distance in the initial population grows for increasing n. For n small all individuals generated are grouped together; for larger n the individuals are more spread, up to a critical n after which the individuals become maximally distant on average. We set n such that the average SHD in the initial population is half of the diameter of the space (i.e., SHD=0.5). We refer to the GNMA-GP with the random-walk initialisation as GNMA-RW.

All the experiments used standard parameters for the classic Nelder-Mead Algorithm, which are $\alpha = 1$, $\gamma = 2$, $\rho = 0.5$ and $\sigma = 0.5$. As baselines for comparison we used (1) a volatile GP setting with standard subtree crossover (50%) and mutation (50%), with no reproduction - we call it StdGP; (2) a stable GP setting 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 it HGP. Both used roulette selection and both are elitist in the sense that they always guarantee the survival of the best individual. All experiments used populations of 500 individuals allowed to evolve until 25 thousand evaluations were performed. All except the GNMA-RW populations were initialized with the Ramped Half-and-Half procedure [4] with an initial maximum depth of 8. The initial tree of GNMA-RW was a random full tree of depth 8. The function and terminal sets were the same for both problems, containing eight functions protected according to [4] ($+, -, \times, /, sin, cos, log, exp$) and no constants. Each experiment was repeated 20 times. The plots that follow (except Figure 5) report the median values of these 20 runs.

Figure 2 shows the results obtained in the Regression problem, in terms of best and median fitness, genotypic diversity (percentage of unique individuals in the population) and average SHD between all pairs of individuals. In both GNMA-GP and GNMA-RW there is very premature convergence to suboptimal solutions, as diversity and average SHD drop to minimal values. The random-walk initialization does not make a difference in this problem. Figure 4 shows four examples of the distribution of SHDs in the population along the evolution, where plot c (GNMA-GP in the Regression problem) reveals that the lower values of SHD, in particular SHD=0, dominate from early in the run. A similar behavior is observed for GNMA-RW (shown only for the One-Max-like problem, plot d). On the other hand, both StdGP and HGP keep diversity and average SHD high (higher diversity in StdGP, as expected), dominated by SHD=1 (Figure 4). The median fitness of StdGP (Figure 2b) is not visible due to a high number of bad individuals in the population.

Figure 3 shows analogous results for the One-Max-like problem. In this problem it is GNMA-GP and GNMA-RW that clearly outperform the baselines. Even so, the diversity and average SHD still drop, although not so fast as in the previous problem. Random-walk initialization seems to prevent the loss of diversity to a certain degree, but the lines are very irregular so we look at Figure 5, that shows the evolution of diversity in each of the 20 runs, along with the average. Despite the high variability, it can be observed that for most runs GNMA-RW is able to achieve a higher diversity in the beginning of the run, and maintain it for a longer time. It is not surprising that the median fitness (Figure 3b) is constantly maximal in StdGP, but that may not have been the expectation for HGP. Figure 4b clearly shows that it happens because the SHD distribution is dominated by SHD=1 from the beginning to the end of the evolution. This suggests that HGP could also benefit from random-walk initialization.

Interestingly, when GNMA-GP and GNMA-RW are both successful and not successful, they seem to converge the population too quickly. However, in the two cases this has different causes. As symbolic regression is not a smooth problem w.r.t. SHD, the population shrinks rapidly because most of the attempts by the search operators of finding better solutions by enlarging the simplex fail, and when that happens the default operation is that of shrinking the population. As the one-max-like problem is a smooth problem w.r.t. SHD

[2]This is because in the OneMax problem the fitness of a solution to maximise is the number of ones it has. This is equivalent to a problem in which the fitness of a solution to minimise is given by the Hamming distance from the solution to the string with all bits set to one. For the symmetry of the Hamming space, this problem is in turn equivalent to any problem in which the string with all bits set to one is replaced with a fixed by arbitrary target string.

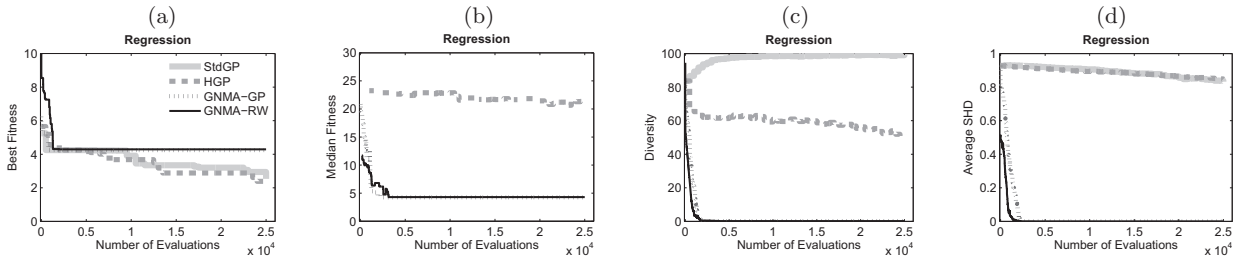

Figure 2: Experimental results on the Regression problem. (a) Best fitness of run; (b) Median fitness of population; (c) Genotypic diversity; (d) Average pairwise SHD in the population.

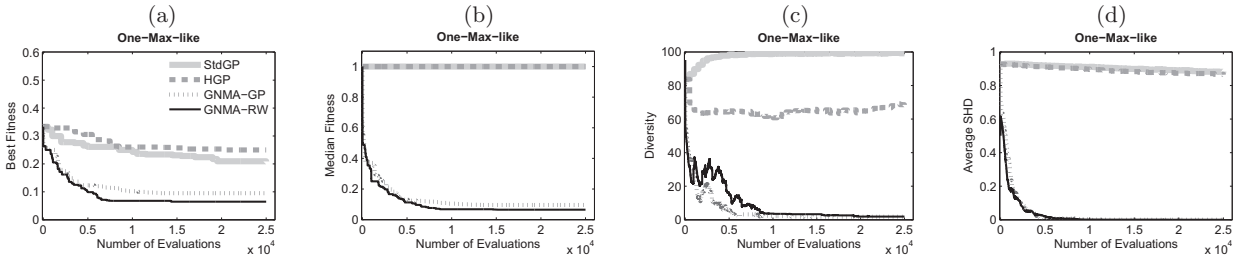

Figure 3: Experimental results on the One-Max-like problem. (a) Best fitness of run; (b) Median fitness of population; (c) Genotypic diversity; (d) Average pairwise SHD in the population.

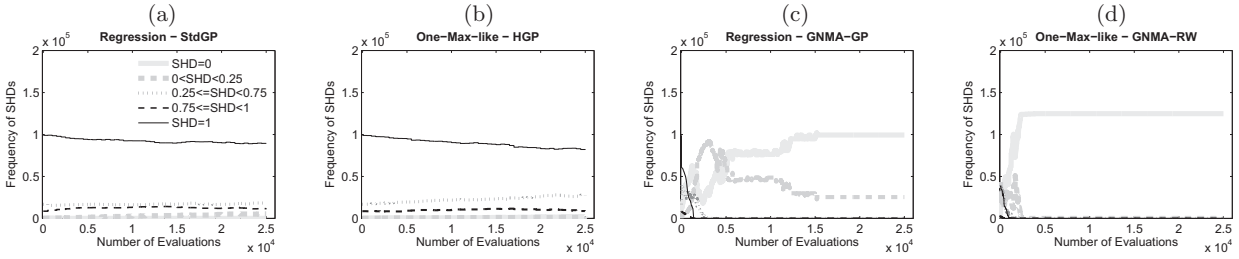

Figure 4: A sample of SHD distributions. (a) Regression, StdGP; (b) One-Max-like, HGP; (c) Regression, GNMA-GP; (d) One-Max-like, GNMA-RW.

by construction, the population moves very rapidly up the gradient, as the NMA search operators are designed to do exactly that, and when the peak is reached the population quickly groups around it, because it cannot find further improvement by attempting enlarging the simplex.

It is also noticeable that the population diversity in the course of a single run can change dramatically, both increasing and decreasing. This feature is related to the search behaviour of the traditional NM on continuous spaces whose span of the simplex adapts to the topography of the fitness landscape searched. However, whereas on continuous spaces the simplex is kept non-degenerate (i.e., the diversity of the population is always maximal) and what adapts is the shape of the simplex, on discrete spaces both shape and diversity are adaptive, due to the discreteness of the space that does not allow the search operators to keep the population diversity always maximal. When using GNMA-GP, the adaptive population diversity is able to partly compensate for the dramatic decrease of diversity due to the degeneracy of the search operators in the initial population. That is why GNMA-GP and GNMA-RW perform similarly.

6. CONCLUSIONS AND FUTURE WORK

In this paper we have demonstrated how to specify the general Geometric Nelder-Mead Algorithm to the space of genetic programs under structural hamming distance. From preliminary experimental results on symbolic regression and on a unimodal problem, the new algorithm, whereas it performs well on the unimodal problem, it does not perform as well as standard GP with swap crossover and with homologous crossover on the symbolic regression problem. This is a rather surprising result as the Geometric Nelder-Mead Algorithm specified to binary strings under Hamming distance performed significantly better than a genetic algorithm on NK-landscapes [10], showing that, in principle, the GNMA may work well when applied to combinatorial spaces. The reason behind it seems to be related with the peculiar geometric properties of the GP space that forces the GNMA towards a degenerate dynamic. It is fair to say that at the

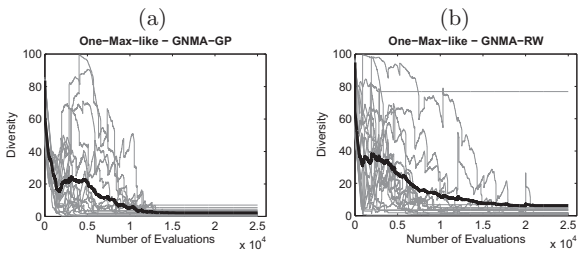

Figure 5: Diversity on the One-Max-like problem, 20 independent runs (grey thin lines) and average (black thick line). (a) Ramped population initialization; (b) Random-walk population initialization.

moment GNMA-GP is still a rather mysterious algorithm. As future work, we will test this new algorithm more thoroughly and on a larger set of problems. Also, we will derive the GNMA for GP programs under other distances which may be more suitable to particular classes of problems, e.g., symbolic regression. Finally, as GNMA is a close relative of GPSO and GDE, we will present the three algorithms in a common theoretical framework highlighting their commonalities and differences and we will compare them experimentally to find out which of their characteristics are better suited to which type of problems and representations.

Acknowledgments

This work was supported by FCT (INESC-ID multiannual funding) through the PIDDAC Program funds. Sara Silva thanks project PTDC/EIA-CCO/103363/2008, FCT, Portugal.

7. REFERENCES

[1] M. Clerc. Discrete particle swarm optimization, illustrated by the traveling salesman problem. In *New Optimization Techniques in Engineering*, pages 219–239. Springer, 2004.

[2] A. Ekart and S. Z. Nemeth. A metric for genetic programs and fitness sharing. In *Genetic Programming, Proceedings of EuroGP'2000*, pages 259–270, 2000.

[3] J. Kennedy and R. C. Eberhart. A discrete binary version of the particle swarm algorithm. *IEEE Transactions on Systems, Man, and Cybernetics*, 5:4104–4108, 1997.

[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-Verlag, 2002.

[6] C. Luo and B. Yu. Low dimensional simplex evolution a hybrid heuristic for global optimization. In *Eighth International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing*, volume 2, pages 470–474, 2007.

[7] A. Moraglio. *Towards a geometric unification of evolutionary algorithms*. PhD thesis, University of Essex, 2007.

[8] A. Moraglio, C. D. Chio, and R. Poli. Geometric particle swarm optimization. In *European Conference on Genetic Programming*, pages 125–136, 2007.

[9] A. Moraglio, C. D. Chio, J. Togelius, and R. Poli. Geometric particle swarm optimization. *Journal of Artificial Evolution and Applications*, 2008:Article ID 143624, 2008.

[10] A. Moraglio and C. Johnson. Geometric generalization of the nelder-mead algorithm. In *Proceedings of the 10th European Conference on Evolutionary Computation in Combinatorial Optimization*, 2010.

[11] A. Moraglio and R. Poli. Geometric landscape of homologous crossover for syntactic trees. In *Proceedings of IEEE congress on evolutionary computation*, pages 427–434, 2005.

[12] 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*, 2007. (to appear).

[13] A. Moraglio and S. Silva. Geometric differential evolution on the space of genetic programs. In *Proceedings of the 13th European Conference on Genetic Programming*, 2010.

[14] A. Moraglio and J. Togelius. Geometric pso for the sudoku puzzle. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 118–125, 2007.

[15] A. Moraglio and J. Togelius. Geometric differential evolution. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, pages 1705–1712, 2009.

[16] J. A. Nelder and R. A. Mead. A simplex method for function minimization. *Computer Journal*, 7:308–313, 1965.

[17] M. O'Neill and A. Brabazon. Grammatical differential evolution. In *Proceedings of the 2006 International Conference on Artificial Intelligence*, pages 231–236. CSREA Press, 2006.

[18] G. Pampara, A. Engelbrecht, and N. Franken. Binary differential evolution. In *IEEE Congress on Evolutionary Computation*, 2006.

[19] K. V. Price, R. M. Storm, and J. A. Lampinen. *Differential Evolution: A Practical Approach to Global Optimization*. Springer, 2005.

[20] T. Takahama and S. Sakai. Constrained optimization by applying the α-constrained method to the nonlinear simplex method with mutations. *IEEE Transactions on Evolutionary Computation*, 9(5):437–451, 2005.

[21] J. Togelius, R. D. Nardi, and A. Moraglio. Geometric pso + gp = particle swarm programming. In *Proceedings of the Congress on Evolutionary Comptutation (CEC)*, 2008.

[22] F. Wang and Y. Qiu. Empirical study of hybrid particle swarm optimizers with the simplex method operator. In *Proceedings of the 5th International Conference on Intelligent Systems Design and Applications*, pages 308–313, 2005.

[23] T. Weise. *Global Optimization Algorithms - Theory and Application*. on-line ebook, 2009.