Runtime Analysis of Mutation-Based Geometric Semantic Genetic Programming for Basis Functions Regression

Alberto Moraglio & Andrea Mambrini
CERCIA, University of Birmingham, Birmingham B15 2TT, UK
{a.moraglio,a.mambrini}@cs.bham.ac.uk

ABSTRACT

Geometric Semantic Genetic Programming (GSGP) is a recently introduced form of Genetic Programming (GP) that searches the semantic space of functions/programs. The fitness landscape seen by GSGP is *always* – for any domain and for any problem – unimodal with a linear slope by construction. This makes the search for the optimum much easier than for traditional GP, and it opens the way to analyse theoretically in a easy manner the optimisation time of GSGP in a *general setting*. Very recent work proposed a runtime analysis of mutation-based GSGP on the class of *all* Boolean functions. We present a runtime analysis of mutation-based GSGP on the class of *all* regression problems with generic basis functions (encompassing e.g., polynomial regression and trigonometric regression).

Categories and Subject Descriptors

F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems

Keywords

Genetic programming, semantics, geometric crossover, runtime analysis

1. INTRODUCTION

Traditional Genetic Programming searches the space of functions/programs by using search operators that manipulate their syntactic representation, regardless of their actual semantics/behaviour. For instance, subtree swap crossover is used to recombine functions represented as parse trees, regardless of trees representing Boolean expressions, mathematical functions, or computer programs. Although this guarantees that offspring are always syntactically well-formed, it is unclear why such a blind syntactic search should work well for different problems and across domains.

In recent literature, there are a number of approaches that use the semantics of programs in various ways to guide the search of GP [1, 4, 17, 6, 7]. Whereas these semantically aware methods are promising, their implementations are very wasteful as they are heavily based on trial-and-error: search operators are implemented via acting on the syntax of the parents to produce offspring, which are accepted only if some semantic criterion is satisfied. More importantly from a theoretical perspective, these implementations do not provide insights on how syntactic and semantic searches relate to each other. Is a *direct* implementation of semantic operators possible? That is, can we act on the syntax of the parent programs and produce offspring that are *guaranteed* to respect some semantic criterion/specification by construction?

Geometric Semantic Genetic Programming [12] is a novel form of genetic programming that answers this question in the affirmative. GSGP uses geometric crossover and geometric mutation [14, 11] to search *directly* the semantic space of functions/programs. Informally, semantic geometric crossover and semantic geometric mutation generate offspring that are guaranteed to be, respectively, "semantically intermediate" and "semantically near" their parents. These search operators can be directly implemented for different domains following a *simple formal recipe* [12], which was used to derive specific forms of GSGP for a number of classic GP domains (i.e, Boolean functions, arithmetic functions and classifiers).

The fitness landscape seen by the semantic geometric operators is *always* unimodal with a linear slope (cone landscape) by construction, as the fitness of an individual is by definition its *semantic distance* to the optimum individual. This suggests that GSGP performs better than standard GP. GSGP was compared with standard GP on several well-known problems across domains (finding Boolean functions, polynomial regressions, and classification tasks) and it consistently found much better solutions with the same budget of fitness evaluations [12]. Furthermore, GSGP has been found more efficient and generalising better than standard GP on some initial studies on real-world problems [2].

Genetic programming has been hard to analyse theoretically. The current literature on GP theory is heterogeneous. Perhaps the most developed theory of GP is the schema theory [8]. There is also some work on Markov models of GP search [10]. There are theory-laden methods to combat bloat based on an exact formalisation of the dynamics of average program size [16]. Other works focus on the analysis of some static structural features of the search space of GP programs (e.g., proportions of programs encoding the same function for different program sizes), and experimental hardness studies of fitness landscapes [8]. There is also some theoretical works on GP from a semantic perspective. In [15], a notion of geometric mutation based on a semantic distance was used to show that the No Free Lunch theorem does not apply to GP. Furthermore, the work [9] analyses traditional subtree crossover in terms of "semantic building blocks" in Boolean functions, reporting that most of the time this crossover does not make useful search in the semantic space.

Runtime analysis is the standard approach to analyse analytically algorithmic performance. In the last decade it has been applied, with an ever increasing success, to randomised search heuristics and it is establishing itself as a leading theory. Despite its success, the analysis is done on a per-algorithm-and-per-problem basis. Obtaining interesting, general runtime results holding on a large class of problems for non-trivial search algorithms would be a major progress. Due to the difficulty of analysing GP, there is only very initial work on its runtime analysis. Durrett et al. [3] present the runtime analysis of a mutation-based GP with a tree representation on very simplified problems (the ORDER and MAJORITY problems).

The cone landscape seen by GSGP makes it very attractive from a theoretical point of view. When GSGP is applied to search the space of Boolean functions, the GSGP search is equivalent to a GA search on OneMax-like problems. Based on this, very recently [13], a runtime analysis of GSGP with various types of mutations on the class of *all* Boolean functions has been possible by simply extending known runtime results for GAs. Remarkably, it was found that GSGP finds the optimum Boolean function in polynomial time on most of the Boolean problems. We continue this line of investigation and present a runtime analysis of GSGP on the class of *all* regression problems with generic basis functions (encompassing e.g., polynomial regression and trigonometric regression). This is a large class of regression problems, which is, however, only a subclass of general symbolic regression that is the natural target class of traditional GP. This work is the first step towards the analysis of more general regression problems.

2. GEOMETRIC SEMANTIC GENETIC PROGRAMMING

A search operator $CX : S \times S \rightarrow S$ is a *geometric crossover* w. r. t. the metric d on S (set of candidate solutions) if for any choice of parents p_1 and p_2, any offspring $o = CX(p_1, p_2)$ is in the segment $[p_1, p_2]$ between parents, i.e., it holds that $d(p_1, o) + d(o, p_2) = d(p_1, p_2)$. A search operator $M : S \rightarrow S$ is a *geometric ϵ-mutation* w. r. t. the metric d if for any parent p, any of its offspring $o = M(p)$ is in the ball of radius ϵ centered in the parent, i.e., $d(o, p) \leq \epsilon$. Given a fitness function $f : S \rightarrow \mathbb{R}$, the geometric search operators induce or see the fitness landscape identified by the triple (f, S, d). Many well-known recombination operators across representations are geometric crossovers [11].

For most applications, genetic programming can be seen as a supervised learning method. Given a training set made of fixed input-output pairs $T = \{(x_1, y_1), ..., (x_N, y_N)\}$ (i.e., fitness cases), a function $h : X \rightarrow Y$ within a certain fixed class H of functions (i.e., the search space specified by the chosen terminal and function sets) is sought that interpolates the known input-output pairs. I.e., for an optimal solution h^* it holds that $\forall (x_i, y_i) \in T : h^*(x_i) = y_i$. The fitness function $F_T : \mathrm{H} \rightarrow \mathbb{R}$ measures the error of a candidate solution h on the training set T. Compared to other learning methods, two distinctive features of GP are (i) it can be applied to learn virtually any type of functions, and (ii) it is a *black-box method*, as it does not need explicit knowledge of the training set, but only of the errors on the training set.

We define the *genotype-phenotype mapping* as the function $P : \mathrm{H} \rightarrow Y^{|X|}$ that maps a representation of a function h (i.e., its genotype) to the vector of the outcomes of the application of the function h to all possible input values in X (i.e., its phenotype), i.e., $P(h) = (h(x_1), ..., h(x_{|X|}))$. We can define a *partial* genotype-phenotype mapping by restricting the set of input values X to a given subset X' as follows: $P_{X'} : \mathrm{H} \rightarrow Y^{|X'|}$ with $P_{X'}(h) = (h(x_1), ..., h(x_{|X'|}))$ with $x_i \in X'$. Let $I = (x_1, ..., x_N)$ and $O = (y_1, ..., y_N)$ be the vectors obtained by splitting inputs and outputs of the pairs in the training set T. The output vector of a function h on the training inputs I is therefore given by its partial genotype-phenotype mapping $P_I(h)$ with input domain restricted to the training inputs I, i.e., $P_I(h) = (h(x_1), ..., h(x_N))$. The training set T identifies the partial genotype-phenotype mapping of the optimal solution h^* restricted to the training inputs I, i.e., $P_I(h^*) = O$.

Traditional measures of error of a function h on the training set T can be *interpreted as distance* between the target output vector O and the output vector $P_I(h)$ measured using some suitable metric D, i.e., $F_T(h) = D(O, P_I(h))$ (to minimise). For example, when the space H of functions considered is the class of Real functions, the input and output spaces are $X = \mathbb{R}^n$ and $Y = \mathbb{R}$, and the output vector restricted to the training inputs is a real vector of size N (i.e., size of training set). A suitable metric D to measure the error as a distance between real vectors is e.g. the Euclidean distance (ED), or also the Manhattan distance (MD).

We define *semantic distance SD* between two functions $h_1, h_2 \in \mathrm{H}$ as the distance between their corresponding output vectors measured with the metric D used in the definition of the fitness function F_T, i.e., $SD(h_1, h_2) = D(P(h_1), P(h_2))$. The semantic distance SD is a genotypic distance induced from a phenotypic metric D, via the genotype-phenotype mapping P [1].

We define *semantic geometric crossover and mutation* as the instantiations of geometric crossover and geometric mutation to the space of functions H endowed with the distance SD. E.g., semantic geometric crossover SGX on Real functions represented e.g. as trees returns offspring Real functions (i.e., trees) such that the output vectors of the offspring are in the Euclidean segment between the output vectors of the parents (w. r .t. all $x_i \in X$).

When the training set covers all possible inputs, the *semantic fitness landscape* seen by an evolutionary algorithm with semantic geometric operators is, from the definition of semantic distance, a unimodal landscape in which the fitness of a solution is its distance in the search space to the optimum. This holds for any domain of application of GP (e.g., Boolean, Arithmetic, Program), any specific problem

[1] P is generally non-injective (i.e., different genotypes may have the same phenotype), but it can be made bijective by interpreting a genotype as a representation of a semantic class (i.e., of all functions with the same semantics) without affecting the subsequent analysis (see [13]).

within a domain and for any choice of metric for the error function. Naturally, in practice, the training set covers only a fraction of all possible input-output pairs of a function. This has the effect of *adding a particular form of neutrality* to the cone landscape, as only the part of the output vector of a function corresponding to the training set affects its fitness, the remaining large part is "inactive".

GP search with geometric operators w.r.t. the semantic distance SD on the space of functions H (represented e.g. as trees) is formally equivalent to EA search with geometric operators w.r.t. the distance D on the space of output vectors. This is because: (i) semantic classes of functions are in bijective correspondence with output vectors, as "functions with the same output vector" is the defining property of a semantic class of function; (ii) semantic geometric operators on functions (i.e., trees) are isomorphic to geometric operators on output vectors, as SD is induced from D via the genotype-phenotype mapping P (see also diagram (1) and explanation in the next section). Despite this formal equivalence, actually encoding a function in an EA using its output vector instead of, say, a parse tree, is futile: in the end we want to find a function represented in an *intensive form* (i.e., as a tree) that can represent concisely "interesting" functions and that allows for meaningful generalisation of the training set.

The commutative diagram below illustrates the relationship between the semantic geometric crossover GX_{SD} on genotypes (e.g., trees) on the top, and the geometric crossover (GX_D) operating on the phenotypes (i.e., output vectors) induced by the genotype-phenotype mapping P, at the bottom. It holds that for any $T1, T2$ and $T3 = GX_{SD}(T1, T2)$ then $P(T3) = GX_D(P(T1), P(T2))$.

$$\begin{tikzcd}
T1 \arrow[d, "P"] & \times & T2 \arrow[d, "P"] \arrow[rr, "GX_{SD}"] & & T3 \arrow[d, "P"] \\
O1 & \times & O2 \arrow[rr, "GX_D"] & & O3
\end{tikzcd} \quad (1)$$

The problem of finding an algorithmic characterization of semantic geometric crossover can be stated as follows: given a family of functions H, find a recombination operator GX_{SD} (unknown) acting on elements of H that induces via the genotype phenotype mapping P a geometric crossover GX_D (known) on output vectors. E.g., for the case of Real functions with fitness measure based on Euclidean distance, output vectors are real vectors and GX_D is a line crossover that returns offspring vectors on the (Euclidean) line segment between parent vectors. We want to derive a recombination operator acting on Real functions that corresponds to a line crossover on their output vectors. Note that there is a different type of semantic geometric crossover for each choice of space H and distance D. Consequently, there are different semantic crossovers for different GP domains.

DEFINITION 1. *Given two parent functions $T1, T2 : \mathbb{R}^n \rightarrow \mathbb{R}$, the recombinations SGXE and SGXM return the real function $T3 = (T1 \cdot TR) + ((1 - TR) \cdot T2)$ where TR is a random real constant in $[0, 1]$ (SGXE) (see Fig. 1), or a random real function with codomain $[0, 1]$ (SGXM). Given a parent function $T : \mathbb{R}^n \rightarrow \mathbb{R}$, the mutation SGMR with mutation step ms returns the real function $TM = T + ms \cdot (TR1 - TR2)$ where $TR1$ and $TR2$ are random real functions.*

THEOREM 1. *SGXE and SGXM are semantic geometric crossovers for the space of real functions with fitness function based on Euclidean and Manhattan distances, respectively, for any training set and any real problem. SGMR is a semantic ϵ-geometric mutation for real functions with fitness function based on Euclidean and Manhattan distances. The mean of its probability distribution is the parent, and ϵ is proportional to the step ms.*

Table 1: Training set for the polynomial regression problem (first 2 columns). Output vectors of trees in Figure 1 (last 4 columns): of parents T1 and T2, of random value TR, and of offspring T3.

X	Y	$P(T1)$	$P(T2)$	$P(TR)$	$P(T3)$
-1	1	0	0	0.5	0
-0.5	0.75	0.5	-0.75	0.5	-0.125
0	1	1	-1	0.5	0
+0.5	1.75	1.5	-0.75	0.5	0.375
+1	3	2	0	0.5	1

The proof of the previous theorem can be found in [12]. In the following, we give an example to illustrate the theorem for the SGXE crossover. Let us consider the simple symbolic regression problem, in which, we want to find an expression whose values match those on the quadratic polynomial $x^2 + x + 1$ in the range $[-1, +1]$. Let us say the target function values for $x \in \{-1, -0.5, 0, +0.5, +1\}$ are given as training set (see two leftmost columns of Table 1, X inputs and Y outputs). The target output vector Y is the real vector $(1, 0.75, 1, 1.75, 3)$ (column 2 of Table 1). For each tree representing a real function, one can obtain its output vector by querying the tree on the inputs X. The output vectors of the trees in Figure 1 are in the last 4 columns of Table 1. The fitness $f(T)$ of a tree T (to minimise) is the Euclidean distance between its output vector $P(T)$ and the target output vector Y (restricted to the outputs of the training set), e.g., the fitness of parent $T1$ is $f(T1) = ED(P(T1), Y) = ED((0, 0.5, 1, 1.5, 2), (1, 0.75, 1, 1.75, 3)) \simeq 1.436$. The semantic distance between two trees $T1$ and $T2$ is the Euclidean distance between their output vectors $P(T1)$ and $P(T2)$, e.g., the semantic distance between parent trees $T1$ and $T2$ is $SD(T1, T2) = ED(P(T1), P(T2)) = ED((0, 0.5, 1, 1.5, 2), (0, -0.75, -1, -0.75, 0)) \simeq 3.824$. Let us now consider the relations between the output vectors of the trees in Table 1. This crossover on output vectors is a geometric crossover w.r.t. Euclidean distance, as $P(T3)$ is in the Euclidean segment between $P(T1)$ and $P(T2)$ as $P(T3)$ is obtained as a convex combination of $P(T1)$ and $P(T2)$. We can also verify this on the example using the distance relation for the line segment:
$ED((0, 0.5, 1, 1.5, 2), (0, -0.125, 0, 0.125, 1)) +$
$ED((0, -0.125, 0, 0.125, 1), (0, -0.75, -1, -0.75, 0)) =$
$1.912 + 1.912 = 3.824 =$
$ED((0, 0.5, 1, 1.5, 2), (0, -0.75, -1, -0.75, 0))$.
This shows that the crossover on trees in Figure 1 is a semantic geometric crossover w.r.t. Euclidean distance.

As the syntax of the offspring of semantic operators contain at least one parent, the size of individuals grows quickly in the number of generations. To keep their size manageable during evolution, we need to *simplify algebraically* offspring sufficiently and efficiently *without changing the computed*

```
          +                   Crossover Scheme                   Offspring
T1 =    /   \
       X     1
                                      +                               +
             -                      /     \                         /     \
           /   \                  *         *                     *         *
T2 =   *        1     T3 =     /   \     /   \             =    /   \    /   \
        /   \                 T1   TR  T2    -                X   0.5  0.5   *
       X     X                              /  \                            /   \
                                           1    TR                         X     X

TR =   0.5
```

Figure 1: T1 and T2 are parent functions and TR is a random value in $[0, 1]$. The offspring T3 is obtained by substituting T1, T2 and TR in the crossover scheme and simplifying algebraically.

function. The search of semantic crossover and semantic mutation is unaffected by the simplification, which can then be done at any moment and in any amount. In practice, we need to work with "well-behaved" families of functions whose offspring can be easily simplified. Furthermore, as the semantic operators generate offspring as a functional combination of their parents, *it does not matter how functions are actually represented* (e.g., trees, graphs, sequences) as the representation does not affect the search behaviour. We have represented functions as trees only to contrast this framework with traditional GP. The following analysis focuses on linear combinations of basis functions as family of functions, which can be represented as fixed-length vectors of coefficients, and GP with mutation only.

3. RUNTIME ANALYSIS

3.1 Problem Definition

The problem class we consider is a general form of regression – basis functions regression – defined as follows. Let $g_j : \mathbb{R}^n \to \mathbb{R}$ be a family of m basis functions, and H be the class of functions expressible as linear combination of the basis functions $\{g_j\}$. Let $p : \mathbb{R}^n \to \mathbb{R}$ be an unknown target function, and $T = \{(i_1, o_1), ..., (i_k, o_k)\}$ be a fixed set of input/output pairs of p. The aim is to find a function $h \in H$ that best approximates p on the training examples. I.e., $h^* = argmin_{h \in H}\{ED(P_I(h), O)\}$. By using different families of basis functions, one obtains well-known specific regression classes such as simple linear regression, polynomial regression, splines regression, trigonometric regression and more. E.g., for polynomial regression with a single input variable, one can choose $g_j = x^j$.

The general problem class can be solved directly, without search, using the *least squares method*. A function $h \in H$ can be written as $h(x_1, \dots, x_n) = c_1 \cdot g_1(x_1, \dots, x_n) + \dots + c_m \cdot g_m(x_1, \dots, x_n)$ and it is identified in H, given $\{g_j\}$, by the vector of coefficients $C = (c_1, \dots, c_m)$. The optimal vector of coefficients can be determined solving the system of simultaneous linear equations obtained by using the training set T, i.e., $o_z = c_1 \cdot g_1(i_z) + \dots + c_m \cdot g_m(i_z)$ with $z = 1, \dots, k$. In matrix form is $O' = GC'$ with O is the vector of training set outputs and G is the matrix $\{g_j(i_z)\}_{j,z}$. The solution is $C' = G^+O'$ where G^+ is the Moore-Penrose pseudoinverse of G that is a generalisation of inverse matrix that returns a well-defined result when the system is underdetermined ($m > k$) or overdetermined ($m < k$). The optimal approximating function so computed *minimises the Euclidean distance* to the target output vector.

Symbolic regression is more flexible and general than basis functions regression as it allows to search variable length functions and functions obtained by arbitrary functional compositions of a base set of functions, not only linear combinations. Furthermore, traditional GP solves symbolic regression problems in a black-box setting: it does not use the knowledge of the training set, but only the errors of candidate solutions on the training set. GSGP goes beyond the least square method as it can solve the basis function regression problem in the black-box setting (Sections 3.3 and 3.4), and it can search the space of variable length linear combinations of basis functions.

3.2 Mutation Design

In [5], a runtime analysis of 1+1-ES with adaptive isotropic Gaussian mutation on the sphere function is reported showing that 1+1-ES is efficient on the sphere. This analysis applies unchanged to the Euclidean cone landscape.

We *reduce* the runtime analysis of GSGP for *any* basis functions regression problem, i.e., any choice of basis functions g_j, any choice of function to approximate p, and any choice of training set T, to the above settings. This is achieved by designing a sematic mutation operator for real functions that on the output vector space *always* corresponds to an isotropic Gaussian mutation.

The semantic mutation we consider is of the form $o(x) = p(x) + ms \cdot r(x)$ where $p(x) \in H$ is the parent function, $ms \in \mathbb{R}^+$ is the mutation step, $r(x)$ is the perturbing function which is sampled according to some probability distribution over H. As $o(x)$ is a linear combination of elements of H it is also in H. For a fixed number of basis functions m, we can represent functions in H by their (fixed-length) vectors of coefficients in the linear combination. The semantic mutation on this representation becomes $c_o = c_p + ms \cdot c_r$ where $c_o, c_p, c_r \in \mathbb{R}^m$ denote the vectors of coefficients of $o(x)$, $p(x)$ and $r(x)$. Simplification of the offspring in this representation is implicitly achieved by algebraic operations on real vectors. The genotype-phenotype mapping on this representation becomes a function $P : \mathbb{R}^m \to \mathbb{R}^k$ that maps vectors of coefficients to output vectors.

LEMMA 1. *The genotype-phenotype mapping P is a linear map. So, it holds that $P(c_1 + c_2) = P(c_1) + P(c_2)$ and $P(\lambda c) = \lambda P(c)$ for all $c_1, c_2, c \in \mathbb{R}^m$ and $\lambda \in \mathbb{R}$.*

PROOF. The mapping P from C to O can be expressed in matrix form as $O' = GC'$ (see Section 3.1). From linear algebra, we know that this is a linear map. $\square$

Consequently, line segments and circles on the space of vectors of coefficients (genotypes) are projected via P to, respectively, line segments and (possibly rotated and rescaled) ellipses on the output vectors space (phenotypes). To obtain a circle on phenotype space, one needs an ellipse in the genotype space that compensates for the transformation P. In terms of semantic operators, this means that a line crossover on genotypes induces a line crossover on phenotypes. A carefully chosen non-isotropic Gaussian mutation on the space of genotypes corresponds to an isotropic Gaussian mutation on the space of phenotypes. We prove the last statement formally.

THEOREM 2. *The non-isotropic Gaussian mutation on genotypes $c_o = c_p + ms \cdot c_r$ with $c_r \sim N(0, G^+IG'^+)$ induces the isotropic Gaussian mutation on phenotypes $P(c_o) = P(c_p) + ms \cdot P(c_r)$ with $P(c_r) \sim N(0, I)$.*

PROOF. $c_o = c_p + ms \cdot c_r$; $P(c_o) = P(c_p + ms \cdot c_r)$; For linearity of P: $P(c_o) = P(c_p) + ms \cdot P(c_r)$; $P(c_r)' = Gc_r'$ is a linear transformation of the multivariate Gaussian c_r: $P(c_r) \sim N(0, G \cdot G^+IG'^+ \cdot G') = N(0, I \cdot I \cdot I') = N(0, I)$ □

The implementation of the semantic mutation requires to sample c_r from $N(0, G^+IG'^+)$. This can be done by multiplying G^+ by a vector formed by sampling k times the normal distribution $N(0, 1)$.

3.3 Analysis with Known Training Inputs

In [5], an asymptotic runtime analysis to obtain an approximated solution within ϵ from the optimum [2] of the standard (1+1)-ES with the 1/5-rule (Algorithm 1) on the Euclidean cone is reported. It was found that it takes $\Theta(n)$ generations (where n is the dimension of the search space) to halve the distance $|c|$ from the initial point c to the optimum. Consequently, it takes $\Theta(n \log_2(\frac{|c|}{\epsilon}))$ generations to find an approximated solution within ϵ from the optimum. When the search space is a hyper-box with length of the sides constant in n and ϵ, the maximum distance of the initial point to the optimum is given by the length of the longest diagonal of the hyper-box, which is $\Theta(\sqrt{n})$. The runtime then becomes $\Theta(n \log_2(\frac{\sqrt{n}}{\epsilon}))$. The algorithm is very efficient both in n for any fixed ϵ, as the runtime is $\Theta(n \log n)$, and in $\frac{1}{\epsilon}$ for a fixed n, as the runtime is $\Theta(\log(\frac{1}{\epsilon}))$.

Algorithm 1 (1+1)-ES with 1/5-rule for adaptation

```
1: initialise $curr \in \mathbb{R}^n$ with a starting point
2: $\sigma := f(curr)/n$; $gen := 0$; $succ := 0$
3: while $f(curr) > \epsilon$ do
4:     choose mutation vector $mut \in \mathbb{R}^n$ randomly $\sim N(0, I)$
5:     create offspring vector $off := curr + \sigma \cdot mut$
6:     if $f(off) \leq f(curr)$ then $curr := off$; $succ := succ+1$ endif
7:     if $gen\%n = 0$ then
8:         if $succ < n \cdot 1/5$ then $\sigma := \sigma/2$ else $\sigma := \sigma \cdot 2$ endif
9:         $succ := 0$
10:    end if
11:    $gen := gen + 1$
12: end while
13: return $curr$
```

The GSGP algorithm that corresponds to Algorithm 1 on the space of output vectors can be obtained from it with the following modifications. The search space is the space of functions in H represented by vectors of coefficients, so $curr, mut \in \mathbb{R}^m$, and n is m. The fitness function $f(x)$ returns the Euclidean distance of the output vector of the function x to the target output vector. The mutation vector mut is sampled from $N(0, G^+IG'^+)$. The pseudo-inverse matrix G^+ is needed to sample mut, which is pre-computed in the initialisation of the algorithm. *This algorithm is only partly black-box because it uses knowledge of the inputs of the training set (but not of the outputs) in the computation of G^+.* In the next Section, a complete black-box algorithm is presented.

When the unknown target function p belongs to the search space H or when the number of basis functions m is larger or equal to the number of points in the training set k, GSGP may in the limit find a function h^* that passes through all points of the training set, i.e., $f(h^*) = 0$. When $p \notin H$ and $m < k$, the best approximating function $h^* \in H$ has non-zero fitness on the training set. This fitness value is the smallest error ϵ_{min} GSGP searching H can achieve.

From a runtime viewpoint, the search done by (1+1)-GP is equivalent to the search of (1+1)-ES on the output vector space. The dimension of the space is the size k of the training set, which is therefore a natural definition of problem size. The size of the hyper-box delimiting the output vector space depends on the endpoints of the interval co-domains of the basis functions $\{g_j\}$, the endpoints of the ranges of the coefficients of linear combinations, which we consider both constant w.r.t. k and ϵ, and on the number of basis functions m, which we consider to be some function of k (e.g., $m = k$ to find an interpolating function). The length of the sides of the hyper-box is therefore $\Theta(m)$, and the runtime to find a function h with $f(h) - f(h^*) < \epsilon$ is $\Theta(k \log_2(\frac{m\sqrt{k}}{\epsilon}))$. In particular, the runtime is constant in the number of input variables n of the functions searched.

3.4 Analysis with Random Training Inputs

In Machine Learning, when one can choose the training examples, they are normally selected to cover evenly the input domain X of the functions $h \in H$ so that the distance between any pair of closest training inputs in X is the same (i.e., they form a grid covering the input domain). In this case, the training inputs are known, and the GSGP analysis in the previous section applies. When one cannot choose the training examples, the standard assumption is that they were sampled uniformly at random on the input domain X. We present an analysis for the second case, in which the sampled training inputs are not known (i.e., *full black-box* scenario).

The idea behind the analysis is as follows. As the number of sampled input points grows, their spatial distribution tends to approximate better and better a grid on X. Asymptotically, GSGP with random training inputs behaves as GSGP with a grid of training inputs covering X. The grid can be determined from the geometry of the domain X. Hence, we can *reduce* the asymptotic runtime analysis of GSGP with *unknown* random training inputs to that of GSGP with a *known* grid of points.

3.4.1 Uni-dimensional Case

Let A and B be collections of points of $X = \mathbb{R}$ with $|A| = |B| = n$. We say that the spatial distribution of A approximates that of B within tolerance ϵ if there is a bijective assignment $\sigma : A \to B$ such that the distance be-

[2]In continuous domains, there is zero probability of hitting exactly the optimum.

tween corresponding points via σ is never greater than ϵ, i.e., $\max_{a\in A}|a-\sigma(a)|\leq\epsilon$.

Let us consider $X=[0,1]$. Let $A=\{\frac{1}{n+1},\frac{2}{n+1},\ldots,\frac{n}{n+1}\}$ be a grid on X and B be a set of n points sampled uniformly at random in X. Let σ be the assignment obtained by ordering B in ascending order and pairing its elements with elements of A also considered in ascending order, i.e., the smallest element in B corresponds to $\frac{1}{n+1}$, the second smallest to $\frac{2}{n+1}$, and so on.

The elements of A have been chosen to coincide with the expected values of the order statistics of B. The order statistics of B are the random variables $b(1),b(2),\ldots,b(n)$ obtained by ordering in ascending order the independent random variables $b_1,b_2,\ldots,b_n\sim U([0,1])$. E.g., $b(1)=\min\{b_1,b_2,\ldots,b_n\}$ and $b(n)=\max\{b_1,b_2,\ldots,b_n\}$. As n grows, the order statistics concentrate towards their expected values. This means that uniformly sampled points become closer and closer to the points of the grid for larger n. We make this precise.

Theorem 3. *The random points B are approximated by the grid A within an arbitrarily small error $\epsilon>0$ with overwhelming probability w.r.t. n.*

Proof. The order statistics of B are $b(k)\sim Beta(k,n-k+1)$ for $k=1\ldots n$, with means $m(k)=\frac{k}{n+1}$. The cdf of $Beta(\alpha,\beta)$ is the regularised incomplete beta function $I(x;\alpha,\beta)$.

The grid A approximates the random points B with tolerance $\epsilon>0$, when $|b(k)-m(k)|<\epsilon$ for all $k=1\ldots n$. We want a lower-bound of $P=Pr(|b(k)-m(k)|<\epsilon,\forall k=1\ldots n)$. Note that $b(k)$ are not independent. We can estimate P using the union bound: $P\geq 1-\sum Pr(|b(k)-m(k)|>\epsilon)$.

As $b_i\sim U([0,1])$, the distributions of $b(k)$ and of $b(n-k+1)$ are mirror images of each other w.r.t. the mid-point of $[0,1]$. $\sum_{k=1,n}Pr(|b(k)-m(k)|>\epsilon)=$ $\sum_{k=1,n}(Pr(b(k)>m(k)+\epsilon)+Pr(b(k)<m(k)-\epsilon))=$ $\sum_{k=1,n}Pr(b(k)>m(k)+\epsilon)+\sum_{k=1,n}Pr(b(k)<m(k)-\epsilon)=$ $2\sum_{k=1,n}Pr(b(k)>m(k)+\epsilon)$ by mirror symmetry.

The cdf I is related to the cdf $F(x;q,p)$ of the binomial distribution $Bin(q,p)$ as follows: $F(x;q,p)=1-I(p;x+1,q-x)$. So, $Pr(b(k)>m(k)+\epsilon)=1-I(\frac{k}{n+1}+\epsilon;k,n-k+1)=F(k-1;n,\frac{k}{n+1}+\epsilon)$.

The Hoeffding's inequality gives an upper-bound for the cdf of the binomial: $F(x;q,p)\leq\frac{1}{2}\exp(-2\frac{(qp-x)^2}{q})$.

Hence $Pr(b(k)>m(k)+\epsilon)\leq\frac{1}{2}\exp(-2\frac{(n(\frac{k}{n+1}+\epsilon)-(k-1))^2}{n})$.

Let us write the index $k\in\{1,\ldots,n\}$ as $k=r\cdot n$ with $r\in[0,1]$ a constant w.r.t. n. We have then

$Pr(b(k)>m(k)+\epsilon)\leq\frac{1}{2}\exp(-2\frac{(n(\frac{r\cdot n}{n+1}+\epsilon)-r\cdot n+1))^2}{n})$. For $n\to\infty$, for any k: $Pr(b(k)>m(k)+\epsilon)\leq\frac{1}{2}\exp(-2n\epsilon^2)$. Then $P=1-2\sum_{k=1,n}Pr(b(k)>m(k)+\epsilon)\geq 1-n\exp(-2n\epsilon^2)$ i.e., overwhelming probability of success for any choice of $\epsilon>0$. □

Theorem 4. *With overwhelming probability, the GSGP algorithm with* unknown *uniformly random training inputs with* continuous *basis functions that uses a regular grid as a surrogate for the inputs has the same asymptotic runtime of GSGP with knowledge of the inputs.*

Proof sketch. Let us consider a function F that maps the set of training inputs I to the runtime T of GSGP with the inputs I, i.e., $T=F(I)$. Let I_r be the set of the actual training inputs, and I_g be the surrogate input grid. The runtime of GSGP on these two input sets is $T_r=F(I_r)$ and $T_g=F(I_g)$. As $I_r\to I_g$ w.o.p., then $T_r\to T_g$ w.o.p. provided the function F is continuous. We show that when the basis functions are continuous, F is continuous.

The random inputs I_r converge w.o.p. to the input grid I_g. For the continuity of the basis functions g_j, the matrix $G_r=\{g_j(i_z)\}_{j,z}$ with $i_z\in I_r$ converges w.o.p. to the matrix $G_g=\{g_j(i_z)\}_{j,z}$ with $i_z\in I_g$. For the continuity of the operator of matrix inversion, G_r^{-1} converges w.o.p. to G_g^{-1}. For continuity of matrix product, the mutation vector $mut_r=G_r\cdot rv$ converges w.o.p. to $mut_g=G_g\cdot rv$ with $rv\sim N(0,I)$. The distribution of the offspring $off_r=curr+\sigma\cdot mut_r$ converges w.o.p. to the distribution $off_g=curr+\sigma\cdot mut_g$. The fitness distribution $f(off_r)=ED(G_g\cdot off_r,O)$ converges w.o.p. to the fitness distribution $f(off_g)=ED(G_g\cdot off_g,O)$. Then the probability of improvement over the parent and the expected improvement in fitness also converge. Consequently, the runtime T_r converges w.o.p. to the runtime T_g. □

3.4.2 Multi-dimensional Case

Analogously to the uni-dimensional case, we will choose a grid of points A on the input domain $X=\mathbb{R}^d$, define a bijective assignment σ from A to B, and then show that for any $\epsilon>0$ asymptotically w.o.p. $dist(A,B)=\max_{a\in A}\|a-\sigma(a)\|_\infty\leq\epsilon$.

Given the vectors $a,b\in\mathbb{R}^d$, $\|a-b\|_\infty=\max(|a_1-b_1|,\ldots,|a_d-b_d|)$. Hence $dist(A,B)\leq\epsilon$ is equivalent to requiring the maximum absolute difference *in each coordinate* of all points of A to corresponding points in B is $\leq\epsilon$.

Let us consider $X=[0,1]^d$. Let $|A|=|B|=n^d$ for n integer. Let $A=\{\frac{1}{n+1},\frac{2}{n+1},\ldots,\frac{n}{n+1}\}^d$ be a grid on X and B be a set of points sampled uniformly at random in X.

We create the assignment σ from A to B as follows. We order all points of B in ascending order on the *first coordinate* regardless of the values on the other coordinates. Then we make n groups, putting in group B_1 the first n points in the ordered list, in group B_2 the following n points, and so on. For each group B_i, we order all its points in ascending order on the *second coordinate*. Then we make n subgroups, putting in group $B_{i,1}$ the first n points of B_i, in group $B_{i,2}$ the following n points of B_i, and so on. We continue to form new nested levels of subgroups until all d coordinates have been considered. Now, each group of the last level contains exactly a single point of B, which is uniquely identified by the d-dimensional index of its group. The assignment σ put into correspondence points of B to points of A with the same index, interpreting A as a d-dimensional matrix.

Theorem 5. *The points B tend to the grid A w.o.p.*

Proof. Let us consider a bi-dimensional input space i.e., $d=2$, and let B be a set of $N=n^2$ points sampled uniformly at random in $[0,1]$.

We first determine the joint probability distributions of the coordinates of the ordered points $B_{i,j}$ on $[0,1]^2$. The coordinates of the points in B are i.i.d. r.v. $x,y\sim U([0,1])$. The set of points obtained ordering B on the x-coordinate have their x-coordinates distributed as the order statistics $u_{1:N},u_{2:N},\ldots,u_{N:N}$ of $U([0,1])$, and their y-coordinates i.i.d. $\sim U([0,1])$. The ordered points are then grouped in n groups with the distributions:

$B_1 = \{(u_{1:N}, U([0,1])), \ldots, (u_{n:N}, U([0,1])),$
$B_2 = \{(u_{n+1:N}, U([0,1])), \ldots, (u_{2n:N}, U([0,1])), \ldots,$
$B_n = \{(u_{N-n+1:N}, U([0,1])), \ldots, (u_{N:N}, U([0,1])).$
The points of each group B_i are ordered on the y-coordinate and split in n subgroups obtaining the distributions:
$B_{1,1} = (U(\{u_{1:N}, \ldots, u_{n:N}\}), u_{1:n}), \ldots,$
$B_{1,n} = (U(\{u_{1:N}, \ldots, u_{n:N}\}), u_{n:n}),$
$\ldots$
$B_{n,1} = (U(\{u_{N-n+1:N}, \ldots, u_{N:N}\}), u_{1:n}), \ldots,$
$B_{1,n} = (U(\{u_{N-n+1:N}, \ldots, u_{N:N}\}), u_{n:n}).$
Using the law of total probability, the cdf of the compound distribution $U(\{u_{1:N}, \ldots, u_{n:N}\})$ can be determined explicitly: $Pr(U(\{u_{1:N}, \ldots, u_{n:N}\}) \leq z) = \frac{\sum_{i=1,n} Pr(u_{i:N} \leq z)}{n}$, and $E[U(\{u_{1:N}, \ldots, u_{n:N}\})] = \frac{\sum_{i=1,n} E[u_{i:N}]}{n}$.
We can now verify that $E[B_{i,j}] \to A_{i,j}$ for $n \to \infty$.

The means of the distributions $u_{1:N} \ldots u_{n:N}$ are evenly distributed in the interval $[0, 1/n]$ at locations $\frac{1}{n^2+1}, \ldots, \frac{n}{n^2+1}$. These distributions concentrate around their means for growing n in the same way (see proof of Theorem 3). As the interval $[0, 1/n]$ converges towards a single point for $n \to \infty$, the means of $u_{1:N} \ldots u_{n:N}$ converge to the mean of $U(\{u_{1:N}, \ldots, u_{n:N}\})$, and asymptotically $U(\{u_{1:N}, \ldots, u_{n:N}\})$ concentrates around its mean in the same way as $u_{1:N} \ldots u_{n:N}$.

For a given $\epsilon > 0$, the probability that a point in B is not within ϵ from the corresponding point in A in the x-coordinate is $\leq \exp(-2N\epsilon^2) \leq \exp(-2N^{1/2}\epsilon^2)$, and in the y-coordinate is $\leq \exp(-2N^{1/2}\epsilon^2)$. By the union bound, the probability that all points in B are within ϵ in both coordinates is $\geq 1 - 2N\exp(-2N^{1/2}\epsilon^2)$. Generalising the reasoning to d dimensions we obtain a probability of success $\geq 1 - d \cdot N \exp(-2N^{1/d}\epsilon^2)$. $\square$

4. EXPERIMENTS

We have provided analytical asymptotical runtime results for GSGP for the black-box basis functions regression problem. In the following, we compare experimentally GSGP and the standard (1+1)-ES with 1/5-rule for adaptation on the vector of coefficients to search the space of functions (hereinafter (1+1)-ES). These algorithms differ only on the mutation operator used, i.e, GSGP uses the semantic mutation and (1+1)-ES uses the isotropic Gaussian mutation. Unlike GSGP, (1+1)-ES (to search the space of functions) is difficult to analyse theoretically as its mutation operator is distorted by the genotype-phenotype mapping, and it corresponds to a non-isotropic mutation on the space of output vectors. We also compare GSGP, that uses the knowledge of training inputs in the mutation operator, with GSGPg that replaces it with a surrogate regular grid of inputs points covering the input space. The asymptotical runtime of GSGP and GSGPg are the same, but they may be different for finite training set size k. Comparison with standard GP is not included here. In [12], it was shown that (1+1)-ES (to search the space of functions) is much faster than standard GP on polynomial regression.

We test the algorithms on randomly generated problem instances. A random problem p is a polynomial of a single input variable x of degree m with coefficients generated uniformly at random in $[-1, 1]$. A random instance of the problem p is a set T of k input/output pairs obtained by querying p with inputs sampled uniformly at random in $[-1, 1]$. Each point in the graphs in Figure 2 reports the average number of generations of 100 runs, each on a new random instance of a new random problem. The basis functions are powers of x, i.e., $g_i(x) = x^i$ for $i = 0 \ldots m$. We always choose the degree of the polynomial p equal to the number of basis functions.

Figure 2(left) compares GSGP and (1+1)-ES in terms of the number of generations (on the ordinates) to get a solution within tolerance $\epsilon = 0.05$ from the optimum, for increasing size of the training set k (data points for $k = 8, 12, 16, 20$ on the abscissa). The number of basis functions m is set to k. GSGP is much faster and scales in k much better than (1+1)-ES. Standard deviations are not reported on the graph. (1+1)-ES has very large standard deviations, almost equalling the average, GSGP has much smaller standard deviations. This suggests that the performance of (1+1)-ES is much strongly affected by the specific problem instance. The performance curve of GSGP is approximately fitting a $k \log k$, in line with the theoretical analysis.

Figure 2(centre) compares GSGP and (1+1)-ES in terms of the number of generations (on the ordinates) to get increasingly better approximations to the optimum, measured as $\log(\frac{1}{\epsilon})$ on the abscissa (data points for $\epsilon = 1, 0.5, 0.1, 0.05$). The size of the training set k and the number of basis functions m are kept fixed as $m = k = 20$. GSGP scales linearly in $\log(\frac{1}{\epsilon})$, in line with the theory. (1+1)-ES reaches a certain approximation to the optimum quickly, after that it takes increasingly longer time to get better approximations, i.e., its approximation performance does not scale well.

Figure 2(right) shows the effect on the performance of GSGP and GSGPg of the number of basis functions m (data points for $m = 4, 6, 8, 10$) for a fixed training set size $k = 18$, and a fixed approximation ϵ. GSGP and GSGPg have similar performances, as they have asymptotically by the theory, when k is large w.r.t. m. Their performance difference becomes marked as m gets closer to k. However, asymptotically the theory predicts that this effect should vanish as the continuity of the basis functions guarantees the continuity of G. The reason for this discrepancy is not entirely clear, and further experiments are needed to cast light on it.

5. SUMMARY AND FUTURE WORK

Geometric semantic genetic programming is a formal framework to design search operators that act directly on the semantic space of functions. The landscape seen by these operators is *always* a cone by construction. This makes the search potentially more effective than traditional GP, and allows us to easily derive *general* runtime results, an important open challenge.

We have considered GSGP for Basis Functions Regression, which is a large class of regression problems. Problems in this class can be solved directly by the least squares method. However, GSGP can solve them in a black-box setting. We have shown a tight link between GSGP for real functions and Evolutionary Strategy. In particular, we have designed a semantic mutation operator that makes the search of GSGP for *any* basis functions regression problem equivalent to standard (1+1)-ES with isotropic mutation on the sphere function. We could then transfer known runtime results for the (1+1)-ES to GSGP with this semantic mutation, in both partial and complete black-box scenarios. GSGP is very efficient in terms of scalability in both the size of the training set k ($\Theta(k \log k)$) and the target approximation to the optimum ϵ ($\Theta(\log(\frac{1}{\epsilon}))$). Preliminary experiments have shown that GSGP with the new semantic mutation performs well in practice.

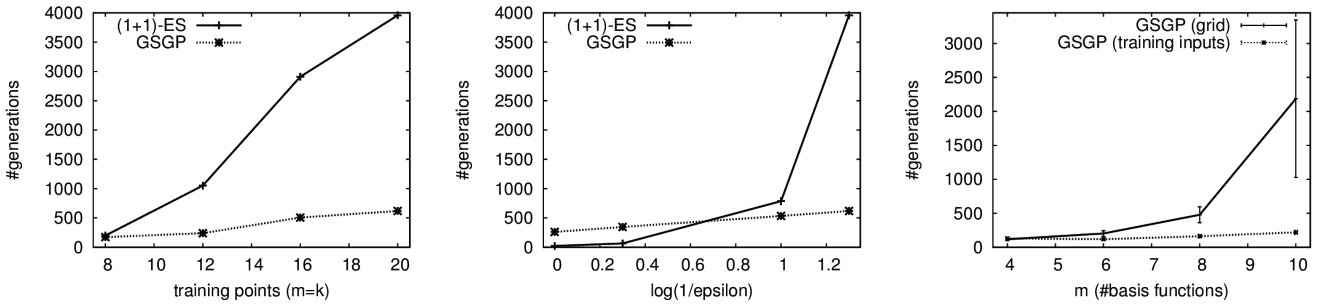

Figure 2: Comparison of GSGP and (1+1)-ES to reach an approximation $\epsilon = 0.05$ (left). Comparison of GSGP and (1+1)-ES with $m = k = 20$ to reach different levels of approximation (centre). Comparison of GSGP using grid and GSGP using the training input for $k = 18$ and different values of m (right).

There is plenty of future work. Two defining characteristics of GP are those of working with variable-length functions, and searching more complex families of functions. We would like to extend this work to variable length linear combinations of basis functions, which is within reach, and explore the possibility of working with more general combinations of functions, like in symbolic regression. Also, we would like to do further experimental investigations of GP with the new semantic operators on standard GP benchmarks, to see how they perform on more practical problems. At present, how functions trained by GP generalise on unseen inputs is a big mystery. As the effect of semantic operators on the output vectors is transparent, this may allow us to explicitly characterise the dependencies between training and testing sets reveling exactly what the inductive bias of GSGP is. Finally, we want to analyse GSGP on other domains. This seems to be within reach as, e.g., semantic operators for classifiers give rise to cone landscapes on integer vectors, which have been studied already for traditional GA, and whose analysis may be extended to GSGP.

6. ACKNOWLEDGMENTS

Alberto Moraglio was supported by EPSRC grant EP/I010297/1.

7. REFERENCES

[1] L. Beadle and C. G. Johnson. Semantic analysis of program initialisation in genetic programming. *Genetic Programming and Evolvable Machines*, 10(3):307–337, 2009.

[2] M. Castelli, L. Manzoni, and L. Vanneschi. An efficient genetic programming system with geometric semantic operators and its application to human oral bioavailability prediction. *arXiv:cs.NE/1208.2437v1*, 2012.

[3] G. Durrett, F. Neumann, and U.-M. O'Reilly. Computational complexity analysis of simple genetic programming on two problems modeling isolated program semantics. In *Workshop on Foundations of Genetic Algorithms*, 2011.

[4] D. Jackson. Phenotypic diversity in initial genetic programming populations. In *Proc. of EuroGP 2010*, pages 98–109, 2010.

[5] J. Jägersküpper. Analysis of a simple evolutionary algorithm for minimization in euclidean spaces. *Theoretical Computer Science*, 379(3):329–347, 2007.

[6] K. Krawiec and P. Lichocki. Approximating geometric crossover in semantic space. In *Proc. of GECCO '09*, pages 987–994, 2009.

[7] K. Krawiec and B. Wieloch. Analysis of semantic modularity for genetic programming. *Foundations of Computing and Decision Sciences*, 34(4):265–285, 2009.

[8] W. Langdon and R. Poli. *Foundations of Genetic Programming*. Springer-Verlag, 2002.

[9] N. F. McPhee, B. Ohs, and T. Hutchison. Semantic building blocks in genetic programming. In *European Conference on Genetic Programming*, 2008.

[10] B. Mitavskiy and J. Rowe. Some results about the markov chains associated to GPs and to general EAs. *Theoretical Computer Science*, 361(1):72–110, 2006.

[11] A. Moraglio. *Towards a Geometric Unification of Evolutionary Algorithms*. PhD thesis, University of Essex, 2007.

[12] A. Moraglio, K. Krawiec, and C. Johnson. Geometric semantic genetic programming. In *Proceedings of Parallel Problem Solving from Nature*, 2012.

[13] A. Moraglio, A. Mambrini, and L. Manzoni. Runtime analysis of mutation-based geometric semantic genetic programming on boolean functions. In *Foundations of Genetic Algorithms*, 2013. (to appear).

[14] A. Moraglio and R. Poli. Topological interpretation of crossover. In *Proc. of GECCO '04*, pages 1377–1388, 2004.

[15] R. Poli, M. Graff, and N. McPhee. Free lunches for function and program induction. In *Workshop on Foundations of Genetic Algorithms*, 2009.

[16] R. Poli and N. F. McPhee. Parsimony pressure made easy: Solving the problem of bloat in gp. In Y. Borenstein and A. Moraglio, editors, *Theory and Principled Methods for Designing Metaheuristics*, chapter 9. Springer, 2012.

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