Theory-Laden design of Mutation-Based Geometric Semantic Genetic Programming for Basis Functions Regression

Alberto Moraglio[1] **Andrea Mambrini**[2]

University of Exeter, UK
University of Birmingham, UK

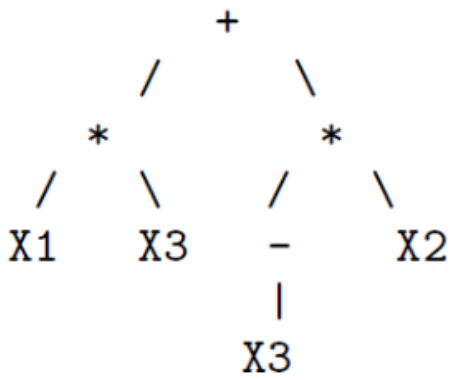
ThRaSH'13
Aberystwyth, UK
14th September 2013

Research question

How can we do **provably good** design of search operators for genetic programming (GP)?

What is the semantic of a program?

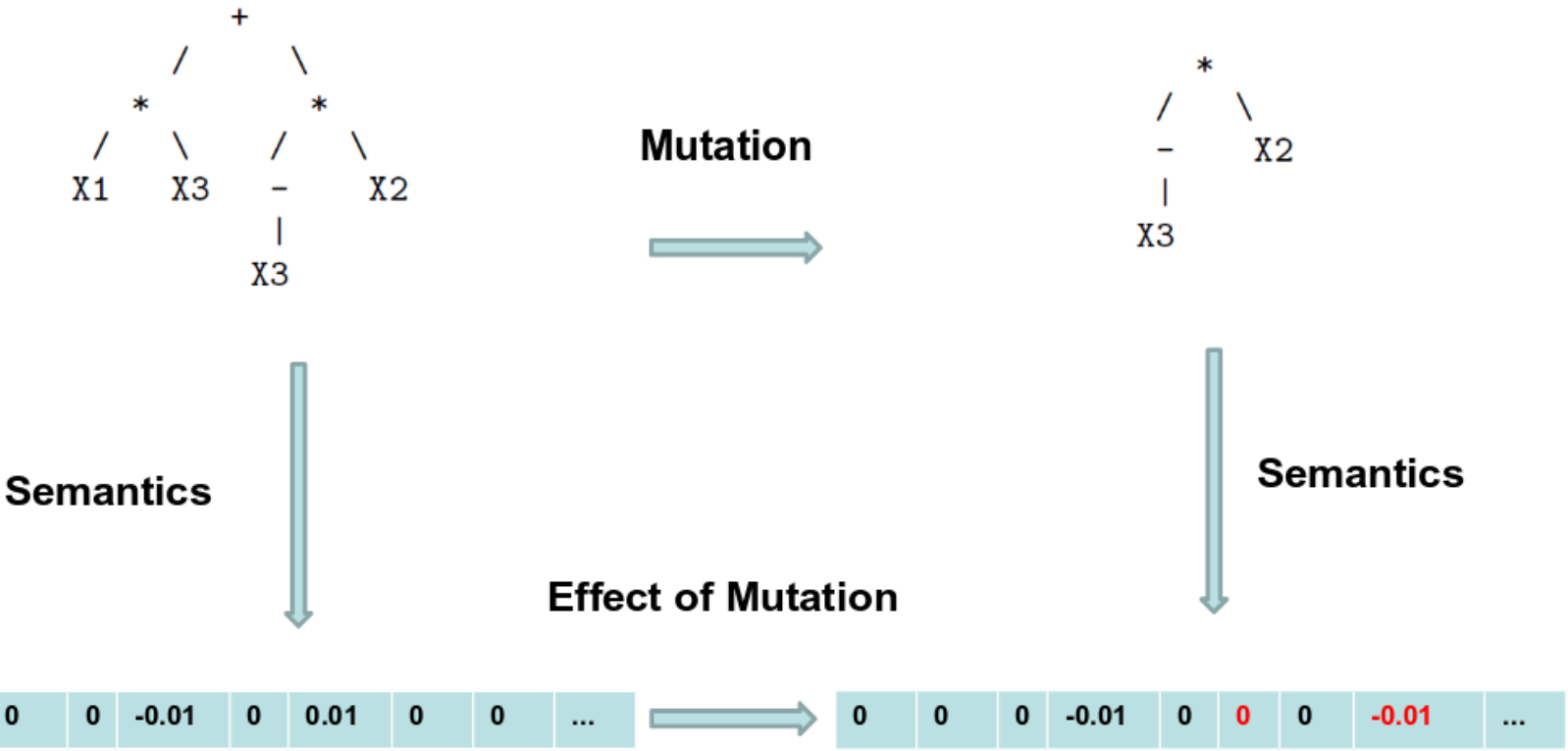
Syntax = Representation



Semantics = Computed Function

X1	X2	X3	Output
0	0	0	0
0	0	0.1	0
0	0.1	0	0
0	0.1	0.1	-0.01
0.1	0	0	0
0.1	0	0.1	0.01
0.1	0.1	0	0
0.1	0.1	0.1	0
...	...	...	...

Semantic effect of search operators



Semantic effect of search operators

- The effect of mutation can be seen as a mutation operator at semantic level.

Semantic effect of search operators

- The effect of mutation can be seen as a mutation operator at semantic level.
- The aim at a semantic level is to match a specific vector (as in Sphere/OneMax)

Semantic effect of search operators

- The effect of mutation can be seen as a mutation operator at semantic level.
- The aim at a semantic level is to match a specific vector (as in Sphere/OneMax)

Can we analyse directly the semantic level?

Semantic Genetic programming

Semantic variation operator

- acts on the syntax of an expression (given a parent expression produce as offspring another expression)
- **guarantee** some semantic features for the offspring (e.g. a semantic mutation might produce an offspring whose output vector always differs just by one element from the parent)

Semantic Genetic programming

Semantic variation operator

- acts on the syntax of an expression (given a parent expression produce as offspring another expression)
- **guarantee** some semantic features for the offspring (e.g. a semantic mutation might produce an offspring whose output vector always differs just by one element from the parent)

How to implement a semantic operator?

Semantic Genetic programming

Semantic variation operator

- acts on the syntax of an expression (given a parent expression produce as offspring another expression)
- **guarantee** some semantic features for the offspring (e.g. a semantic mutation might produce an offspring whose output vector always differs just by one element from the parent)

How to implement a semantic operator?

By trial & error use standard operators and reject offspring that do not conform to the semantic requirement $\rightarrow$ wasteful

Semantic Genetic programming

Semantic variation operator

- acts on the syntax of an expression (given a parent expression produce as offspring another expression)
- **guarantee** some semantic features for the offspring (e.g. a semantic mutation might produce an offspring whose output vector always differs just by one element from the parent)

How to implement a semantic operator?

By trial & error use standard operators and reject offspring that do not conform to the semantic requirement $\rightarrow$ wasteful

By construction : Operate on the syntax of the expression in a way that by construction the semantic criterion is satisfied

Geometric Operators

Geometric operators are defined on the structure of the search space by means of simple geometric shapes

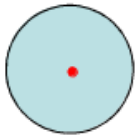
- **Geometric crossover**: all offspring are in the space-specific segment between their parents (*between*)

Geometric Operators

Geometric operators are defined on the structure of the search space by means of simple geometric shapes

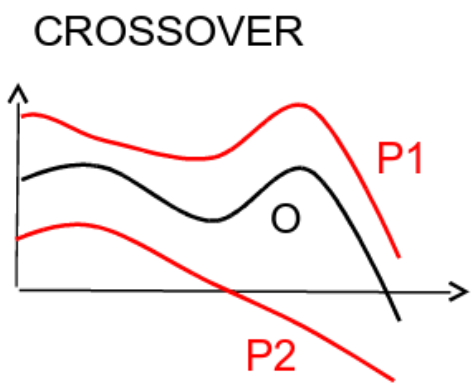
- **Geometric crossover**: all offspring are in the space-specific segment between their parents (*between*)

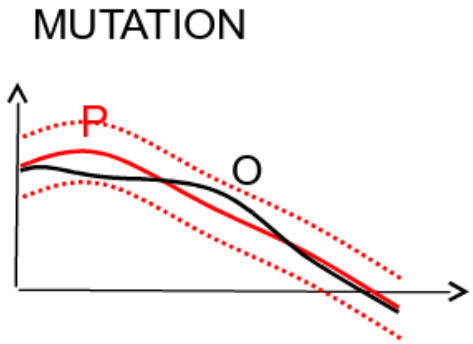
- **Geometric mutation**: offspring are in the space-specific ball of a small radius centred in the parent (*near*)



Geometric Semantic Operators

- **Semantic operators**: act on the syntax of the programs but guarantee that some semantic criterion holds on the output vector
- **Geometric semantic crossover**: all offspring functions are semantically intermediate between their parents
- **Geometric semantic mutation**: offspring functions are semantically near their parent





Search equivalence

Syntax Level

Geometric Semantic GP search for any problem

(actual search on trees)

Semantic Level

Traditional EA search on output vectors on sphere

(theoretical analysis)

Plan: Runtime analysis of GSGP

- **Problem**: Pick a class of problems (i.e **Basis Functions Regression** (BFR))

Plan: Runtime analysis of GSGP

- **Problem**: Pick a class of problems (i.e **Basis Functions Regression** (BFR))
- **Mutation Design**: Design a semantic variation operator such that the search on the output space is equivalent to something known (i.e. **isotropic Gaussian mutation on a real vector**)

Plan: Runtime analysis of GSGP

- **Problem**: Pick a class of problems (i.e **Basis Functions Regression** (BFR))
- **Mutation Design**: Design a semantic variation operator such that the search on the output space is equivalent to something known (i.e. **isotropic Gaussian mutation on a real vector**)
- **Search equivalence**: The search of GP with this mutation for any target function is equivalent to a known algorithm solving a known problem (i.e. **(1+1)-ES on SPHERE**)

Plan: Runtime analysis of GSGP

- **Problem**: Pick a class of problems (i.e **Basis Functions Regression** (BFR))
- **Mutation Design**: Design a semantic variation operator such that the search on the output space is equivalent to something known (i.e. **isotropic Gaussian mutation on a real vector**)
- **Search equivalence**: The search of GP with this mutation for any target function is equivalent to a known algorithm solving a known problem (i.e. **(1+1)-ES on SPHERE**)
- **Reduction to known case**: GP has the same runtime of (1+1)-ES, which is known to ε-approximate the optimum of the sphere efficiently.

Basis Functions Regression (BFR)

- **Given:** k input-output samples $\{(X_1, F(X_1)), \cdots (X_k, F(X_k))\}$ (training set)
- **Evolve:** a function $\widetilde{F}(X)$ from the class H generated by the basis functions $(g_1(x), \ldots, g_m(x))$ which minimize the error on the training set.

Basis Functions Regression (BFR)

- **Given:** k input-output samples $\{(X_1, F(X_1)), \cdots (X_k, F(X_k))\}$ (training set)
- **Evolve:** a function $\widetilde{F}(X)$ from the class H generated by the basis functions $(g_1(x), \ldots, g_m(x))$ which minimize the error on the training set.

Example (polynomial regression):

- Basis functions: $g_1(x) = 1, g_2(x) = x, \ldots, g_m(x) = x^{m-1}$
- Function form: $F(X) = c_1 + c_2 x + c_3 x^2 + \ldots + c_m x^{m-1}$

What are we looking for?

We want an operator, that acting on a parent function $P(X)$ produce an offspring function $P'(X)$ with a predictable behaviour on the output vector $\{P'(X_1), \cdots, P'(X_k)\}$.

What are we looking for?

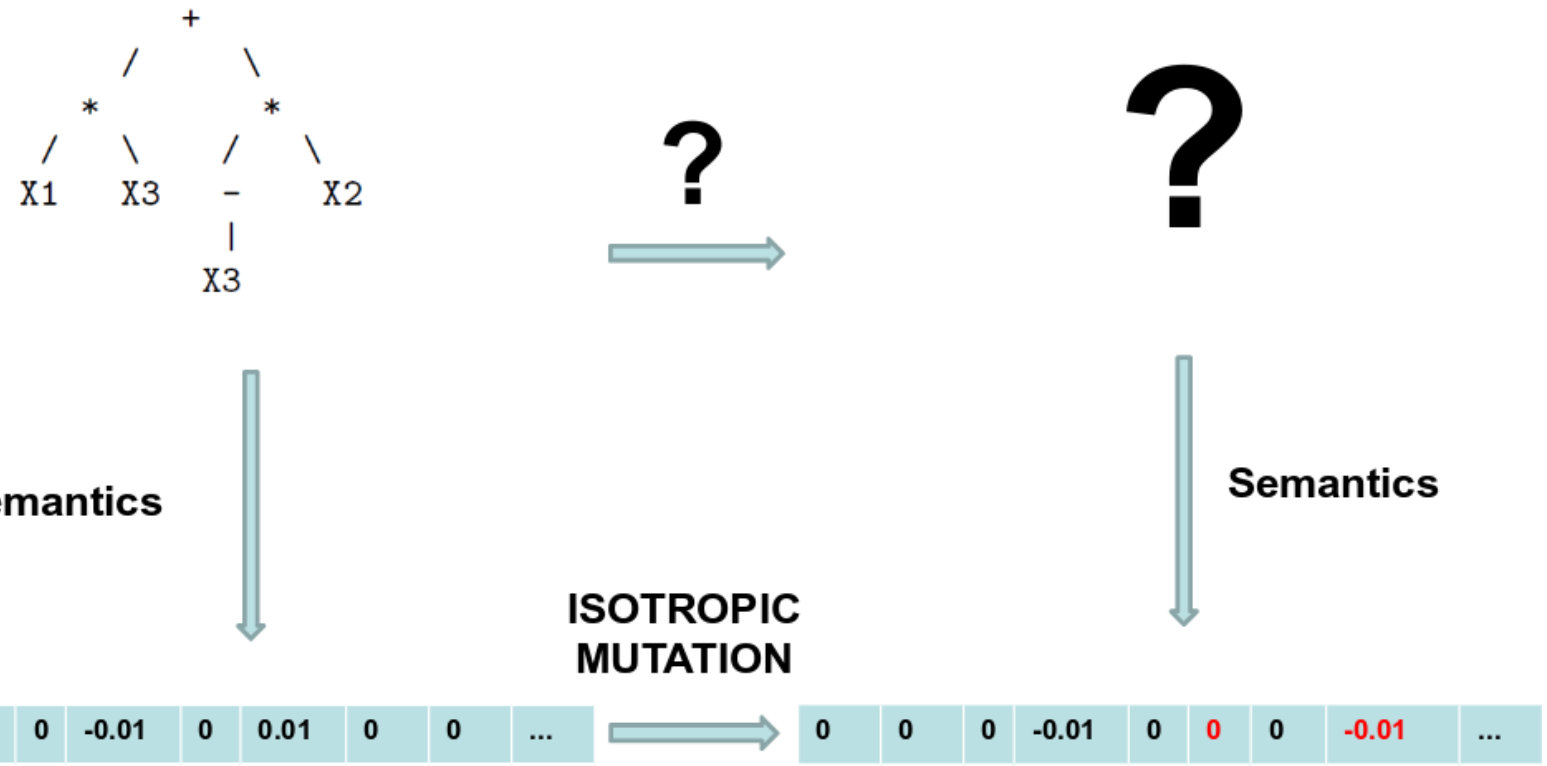
We want an operator, that acting on a parent function $P(X)$ produce an offspring function $P'(X)$ with a predictable behaviour on the output vector $\{P'(X_1), \cdots, P'(X_k)\}$.

- Particularly since the output vector is a real vector we want a semantic operator behaving, on the output vector, as (1+1)-ES.

Algorithm - (1+1)-ES [Evolutionary Strategy]

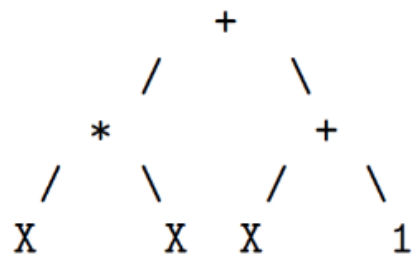
1. Inizialise P_0 with a random real vector $X \in \mathbb{R}^n$
2. Generate a random gaussian vector $R \sim N(\mu, \Sigma)$
3. Create the offspring $X' = X + R$.
4. Select the fittest between X' and X for survival
5. Repeat from point 2 until stopping condition applies

Mutation design



Function representation

TREE



FORMULA

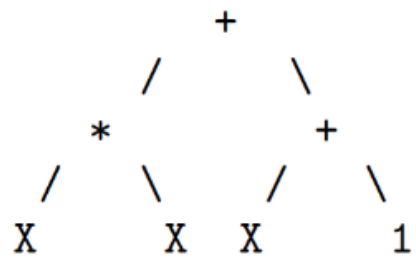
$$x^2 + x + 1$$

VECTOR OF COEFFICIENTS

$$(1,1,1)$$

Function representation

TREE



FORMULA

$$x^2 + x + 1$$

VECTOR OF COEFFICIENTS

$$(1,1,1)$$

Since we evolve functions from the class

$$H = \{c_0 + c_1 g_1(x) + \ldots + c_i g_i(x) + \ldots c_m g_m(x)\}$$

a function is completely identified by its **vector of coefficients**

From coefficient vector to output vector:

- Function: $P(X) = c_1 g_1(x) + \ldots + c_m g_m(x)$
- Coefficient vector $C = (c_1, \ldots c_m)$
- Output vector: $O = (o_1, \ldots, o_k) = (P(X_1), \cdots, P(X_k))$

From coefficient vector to output vector:

- Function: $P(X) = c_1 g_1(x) + \ldots + c_m g_m(x)$
- Coefficient vector $C = (c_1, \ldots c_m)$
- Output vector: $O = (o_1, \ldots, o_k) = (P(X_1), \cdots, P(X_k))$

$$(o_1, \ldots, o_k) = \begin{pmatrix} g_1(X_1) & g_2(X_1) & \cdots & g_m(X_1) \\ g_1(X_2) & g_2(X_2) & & \\ \vdots & \vdots & \ddots & \vdots \\ g_1(X_k) & g_2(X_k) & \cdots & g_m(X_k) \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_m \end{pmatrix}$$

$$O = GC$$

Inducing a specific Gaussian mutation on the output vector

How to perturb the coefficient vector (genotype) of the parent to produce a specific Gaussian mutation on the output vector (phenotype) of the offspring?

Inducing a specific Gaussian mutation on the output vector

How to perturb the coefficient vector (genotype) of the parent to produce a specific Gaussian mutation on the output vector (phenotype) of the offspring?

Theorem:

- If $C' = C + C_r$, with $C_r =\sim N(0, G^{+}\Sigma G'^{+})$
- Then $O'(= GC') = O + O_r$ with $O_r \sim N(0, \Sigma)$

Inducing a specific Gaussian mutation on the output vector

How to perturb the coefficient vector (genotype) of the parent to produce a specific Gaussian mutation on the output vector (phenotype) of the offspring?

Theorem:

- If $C' = C + C_r$, with $C_r =\sim N(0, G^{+}\Sigma G'^{+})$
- Then $O'(= GC') = O + O_r$ with $O_r \sim N(0, \Sigma)$

Proof:

- $O' = GC' = G(C + C_r) = GC + GC_r = O + GC_r$

Inducing a specific Gaussian mutation on the output vector

How to perturb the coefficient vector (genotype) of the parent to produce a specific Gaussian mutation on the output vector (phenotype) of the offspring?

Theorem:

- If $C' = C + C_r$, with $C_r =\sim N(0, G^+\Sigma G'^+)$
- Then $O'(= GC') = O + O_r$ with $O_r \sim N(0, \Sigma)$

Proof:

- $O' = GC' = G(C + C_r) = GC + GC_r = O + GC_r$
- $GC_r \sim N(0, G \cdot G^+\Sigma G'^+ \cdot G') = N(0, I \cdot \Sigma \cdot I') = N(0, \Sigma)$

Inducing a specific Gaussian mutation on the output vector

How to perturb the coefficient vector (genotype) of the parent to produce a specific Gaussian mutation on the output vector (phenotype) of the offspring?

Theorem:

- If $C' = C + C_r$, with $C_r =\sim N(0, G^+ \Sigma G'^+)$
- Then $O'(= GC') = O + O_r$ with $O_r \sim N(0, \Sigma)$

Proof:

- $O' = GC' = G(C + C_r) = GC + GC_r = O + GC_r$
- $GC_r \sim N(0, G \cdot G^+ \Sigma G'^+ \cdot G') = N(0, I \cdot \Sigma \cdot I') = N(0, \Sigma)$
- $\Rightarrow O' = O + O_r$ with $O_r \sim N(0, \Sigma)$

Inducing a specific Gaussian mutation on the output vector

How to perturb the coefficient vector (genotype) of the parent to produce a specific Gaussian mutation on the output vector (phenotype) of the offspring?

Theorem:

- If $C' = C + C_r$, with $C_r =\sim N(0, G^{+}\Sigma G'^{+})$
- Then $O'(= GC') = O + O_r$ with $O_r \sim N(0, \Sigma)$

Proof:

- $O' = GC' = G(C + C_r) = GC + GC_r = O + GC_r$
- $GC_r \sim N(0, G \cdot G^{+}\Sigma G'^{+} \cdot G') = N(0, I \cdot \Sigma \cdot I') = N(0, \Sigma)$
- $\Rightarrow O' = O + O_r$ with $O_r \sim N(0, \Sigma)$

Corollary:

- **If**: Σ is isotropic $[\Sigma = \text{diag}(\sigma, \ldots, \sigma)]$
- **Then**: There is a non-isotropic mutation operation on the coefficients producing isotropic mutation on the output vector

Reduction to runtime analysis of (1+1)-ES on SPHERE

From Jägersküpper 2007[1] we get that:

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

Reduction to runtime analysis of (1+1)-ES on SPHERE

From Jägersküpper 2007[1] we get that:
If:

- Σ is an isotropic diagonal matrix $\Sigma = \text{diag}(\sigma, \ldots, \sigma)$

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

Reduction to runtime analysis of (1+1)-ES on SPHERE

From Jägersküpper 2007[1] we get that:
If:

- Σ is an isotropic diagonal matrix $\Sigma = \text{diag}(\sigma, \ldots, \sigma)$
- σ is adapted throughout the run following the $1/5$ rule, which state to reset σ every k mutation to:
  - $\sigma' = \sigma / c$ if $f > 1/5$
  - $\sigma' = \sigma \cdot c$ if $f < 1/5$

  Where f is the fraction of successful mutation (the ones that actually improve the fitness) and $0.8 < c < 1$.

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

Reduction to runtime analysis of (1+1)-ES on SPHERE

From Jägersküpper 2007[1] we get that:
If:

- Σ is an isotropic diagonal matrix $\Sigma = \text{diag}(\sigma, \ldots, \sigma)$
- σ is adapted throughout the run following the $1/5$ rule, which state to reset σ every k mutation to:
  - $\sigma' = \sigma / c$ if $f > 1/5$
  - $\sigma' = \sigma \cdot c$ if $f < 1/5$

  Where f is the fraction of successful mutation (the ones that actually improve the fitness) and $0.8 < c < 1$.

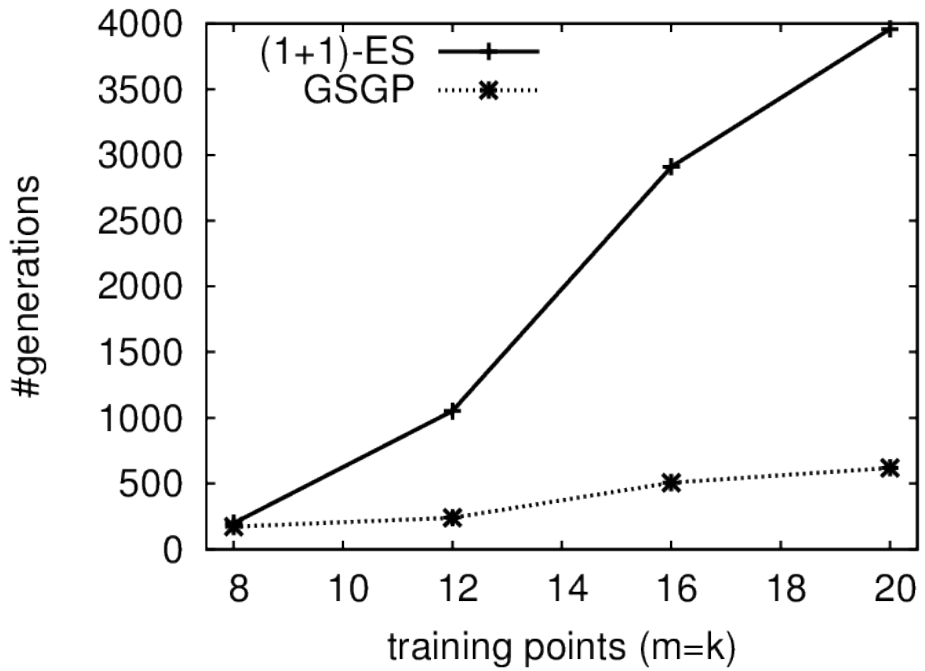
Then:
SPHERE (and then the equivalent GSGP on BFS using the previously designed operator) is ε-approximated in expected time $\Theta(k \log \frac{ck}{\varepsilon})$ (where k is the number of training examples and c is the distance of the first point to the optimum).

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

Experiments

We compared:

- (1+1)-ES searching with an isotropic mutation on the space of coefficients
- GSGP performing non-isotropic mutations on the space of coefficients inducing isotropic mutation on the output vector



More on the paper[2] (black-box scenario)

In order to build the matrix G it is necessary to know the training inputs $(X_1, \ldots, X_k)$.

$$\begin{pmatrix} g_1(X_1) & g_2(X_1) & \cdots & g_m(X_1) \\ g_1(X_2) & g_2(X_2) & & \\ \vdots & \vdots & \ddots & \vdots \\ g_1(X_k) & g_2(X_k) & \cdots & g_m(X_k) \end{pmatrix}$$

[2] A. Moraglio, A. Mambrini – *Runtime Analysis of Mutation-Based Geometric Semantic Genetic Programming for Basis Functions Regressions* – Genetic and Evolutionary Computation Conference (GECCO 2013)

More on the paper[2] (black-box scenario)

In order to build the matrix G it is necessary to know the training inputs $(X_1, \ldots, X_k)$.

$$\begin{pmatrix} g_1(X_1) & g_2(X_1) & \cdots & g_m(X_1) \\ g_1(X_2) & g_2(X_2) & & \\ \vdots & \vdots & \ddots & \vdots \\ g_1(X_k) & g_2(X_k) & \cdots & g_m(X_k) \end{pmatrix}$$

Black-box scenario (input point not known):

- $(X_1, \ldots, X_k)$ are randomly sampled from the input space
- We use a low-discrepancy sequence $(\widetilde{X}_1, \ldots, \widetilde{X}_k)$ as surrogate input point to build $\widetilde{G}$

[2] A. Moraglio, A. Mambrini – *Runtime Analysis of Mutation-Based Geometric Semantic Genetic Programming for Basis Functions Regressions* – Genetic and Evolutionary Computation Conference (GECCO 2013)

More on the paper[2] (black-box scenario)

In order to build the matrix G it is necessary to know the training inputs $(X_1, \ldots, X_k)$.

$$\begin{pmatrix} g_1(X_1) & g_2(X_1) & \cdots & g_m(X_1) \\ g_1(X_2) & g_2(X_2) & & \\ \vdots & \vdots & \ddots & \vdots \\ g_1(X_k) & g_2(X_k) & \cdots & g_m(X_k) \end{pmatrix}$$

Black-box scenario (input point not known):

- $(X_1, \ldots, X_k)$ are randomly sampled from the input space
- We use a low-discrepancy sequence $(\widetilde{X}_1, \ldots, \widetilde{X}_k)$ as surrogate input point to build $\widetilde{G}$

Theorem

For $k \to \infty$, with high probability GSGP using the surrogate matrix $\widetilde{G}$ has the same runtime of GSGP using the exact matrix G.

[2] A. Moraglio, A. Mambrini – *Runtime Analysis of Mutation-Based Geometric Semantic Genetic Programming for Basis Functions Regressions* – Genetic and Evolutionary Computation Conference (GECCO 2013)

Conclusion

What we did:

- **Design**: Designed a GSGP operator performing non-isotropic Gaussian mutation of the coefficients inducing isotropic Gaussian mutation on the output vector.

Conclusion

What we did:

- **Design**: Designed a GSGP operator performing non-isotropic Gaussian mutation of the coefficients inducing isotropic Gaussian mutation on the output vector.
- **Reduction**: The search on the output space is equivalent to (1+1)-ES with isotropic Gaussian mutation on SPHERE.

Conclusion

What we did:

- **Design**: Designed a GSGP operator performing non-isotropic Gaussian mutation of the coefficients inducing isotropic Gaussian mutation on the output vector.
- **Reduction**: The search on the output space is equivalent to (1+1)-ES with isotropic Gaussian mutation on SPHERE.
- **Runtime**: Reusing results from Jägersküpper 2007 we proved that GSGP with the non-isotropic Gaussian mutation on the coefficients solves BFS in expected time $\Theta(k \log \frac{ck}{\varepsilon})$ (k is the number of training examples, c is the distance of the first point to the optimum).

Conclusion

What we did:

- **Design**: Designed a GSGP operator performing non-isotropic Gaussian mutation of the coefficients inducing isotropic Gaussian mutation on the output vector.
- **Reduction**: The search on the output space is equivalent to (1+1)-ES with isotropic Gaussian mutation on SPHERE.
- **Runtime**: Reusing results from Jägersküpper 2007 we proved that GSGP with the non-isotropic Gaussian mutation on the coefficients solves BFS in expected time $\Theta(k \log \frac{ck}{\varepsilon})$ (k is the number of training examples, c is the distance of the first point to the optimum).

Take home messages:

Conclusion

What we did:

- **Design**: Designed a GSGP operator performing non-isotropic Gaussian mutation of the coefficients inducing isotropic Gaussian mutation on the output vector.
- **Reduction**: The search on the output space is equivalent to (1+1)-ES with isotropic Gaussian mutation on SPHERE.
- **Runtime**: Reusing results from Jägersküpper 2007 we proved that GSGP with the non-isotropic Gaussian mutation on the coefficients solves BFS in expected time $\Theta(k \log \frac{ck}{\varepsilon})$ (k is the number of training examples, c is the distance of the first point to the optimum).

Take home messages:

- GSGP makes the search performed by GP on the space of functions (trees) equivalent to that performed by a GA on the output space (real/integer vectors).

Conclusion

What we did:

- **Design**: Designed a GSGP operator performing non-isotropic Gaussian mutation of the coefficients inducing isotropic Gaussian mutation on the output vector.
- **Reduction**: The search on the output space is equivalent to (1+1)-ES with isotropic Gaussian mutation on SPHERE.
- **Runtime**: Reusing results from Jägersküpper 2007 we proved that GSGP with the non-isotropic Gaussian mutation on the coefficients solves BFS in expected time $\Theta(k \log \frac{ck}{\varepsilon})$ (k is the number of training examples, c is the distance of the first point to the optimum).

Take home messages:

- GSGP makes the search performed by GP on the space of functions (trees) equivalent to that performed by a GA on the output space (real/integer vectors).
- It is thus possible to easily get runtime results for GP reusing runtime results for the GA

Conclusion

What we did:

- **Design**: Designed a GSGP operator performing non-isotropic Gaussian mutation of the coefficients inducing isotropic Gaussian mutation on the output vector.
- **Reduction**: The search on the output space is equivalent to (1+1)-ES with isotropic Gaussian mutation on SPHERE.
- **Runtime**: Reusing results from Jägersküpper 2007 we proved that GSGP with the non-isotropic Gaussian mutation on the coefficients solves BFS in expected time $\Theta(k \log \frac{ck}{\varepsilon})$ (k is the number of training examples, c is the distance of the first point to the optimum).

Take home messages:

- GSGP makes the search performed by GP on the space of functions (trees) equivalent to that performed by a GA on the output space (real/integer vectors).
- It is thus possible to easily get runtime results for GP reusing runtime results for the GA
- We can use those theoretical results to design good semantic GP operators

Conclusion

What we did:

- **Design**: Designed a GSGP operator performing non-isotropic Gaussian mutation of the coefficients inducing isotropic Gaussian mutation on the output vector.
- **Reduction**: The search on the output space is equivalent to (1+1)-ES with isotropic Gaussian mutation on SPHERE.
- **Runtime**: Reusing results from Jägersküpper 2007 we proved that GSGP with the non-isotropic Gaussian mutation on the coefficients solves BFS in expected time $\Theta(k \log \frac{ck}{\varepsilon})$ (k is the number of training examples, c is the distance of the first point to the optimum).

Take home messages:

- GSGP makes the search performed by GP on the space of functions (trees) equivalent to that performed by a GA on the output space (real/integer vectors).
- It is thus possible to easily get runtime results for GP reusing runtime results for the GA
- We can use those theoretical results to design good semantic GP operators

Thank you!