Exploring the Impact of Data Scale on Mutation Step Size in SLIM-GSGP

Davide Farinati[1], Gloria Pietropolli[2], and Leonardo Vanneschi[1]

[1] NOVA Information Management School (NOVA IMS), Universidade Nova de Lisboa, Campus de Campolide, 1070-312 Lisboa, Portugal {dfarinati, lvanneschi}@novaims.unl.pt

[2] Department of Mathematics, Informatics, and Geosciences, University of Trieste, Italy. gloria.pietropolli@units.it

Abstract. The Semantic Learning algorithm based on Inflate and deflate Mutation (SLIM) is a promising recent variant of Geometric Semantic Genetic Programming (GSGP) that introduces a new Deflate Geometric Semantic Mutation (DGSM). This operator maintains the key feature of the standard Geometric Semantic Mutation (GSM), inducing a unimodal error surface for any supervised learning problem, while generating smaller offspring than their parents, and thus allowing SLIM to generate compact, and potentially interpretable, final solutions. A key parameter controlling the evolution process in both GSGP and SLIM is the Mutation Step (MS), which regulates the extent of perturbation to the parent semantics. While it is intuitive that the optimal value of MS has a relationship with the scale of the dataset features, to the best of our knowledge no prior research has extensively explored this relationship. In this work, we provide the first comprehensive investigation into this topic. First, we hypothesize a general rule by analyzing results from artificial datasets, and then we confirm these findings with more complex, real-world datasets. This approach offers a solid alternative to the typical hyperparameter tuning approach.

Keywords: Genetic Programming · Geometric Semantic Genetic Programming · Geometric Mutation · Mutation Step · Symbolic Regression.

1 Introduction

Geometric Semantic Genetic Programming (GSGP) is a variant of Genetic Programming (GP) [16, 26] that uses Geometric Semantic Operators (GSOs) to replace the traditional (syntax-based) crossover and mutation and induce precise and known geometric properties in the semantic space [20, 34]. The key property of GSOs is that they induce a unimodal error surface for any supervised learning problem. However, GSGP is also known for producing extremely large final solutions, that are usually not interpretable by humans. A new variant of GSGP was recently introduced, which preserves the property of inducing a unimodal error surface, while also producing models that are compact enough to be interpreted by humans [30, 33]. This variant, called Semantic Learning algorithm based on

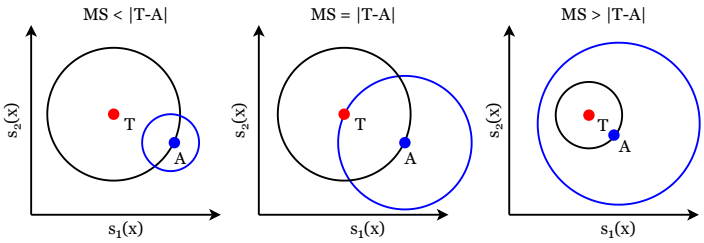


Fig. 1: Visual representation of the impact of the MS in a bidimensional feature space($s_1(x)$ and $s_2(x)$). The target point T is shown, with the black circle representing the space of solutions fitter than an individual, whose semantic is point A. The blue circle illustrates the space of solutions that can be generated by mutating that individual. Three conditions are shown: when the MS is smaller than, equal to, or greater than the distance between the target (T) and the individual.

Inflate and deflate Mutation (SLIM), relies on two types of mutation: besides the traditional Geometric Semantic Mutation (GSM) of GSGP, called Inflate Geometric Semantic Mutation (IGSM) as it creates offspring are larger than their parents, SLIM introduces a Deflate Geometric Semantic Mutation (DGSM), so called because of its property of producing offspring smaller than their parents. A crossover for SLIM has not been defined yet and a significant amount of literature indicates that GSGP using only mutation is often competitive with, or even better than, GSGP using both crossover and mutation [4, 21, 35]. For these reasons, we will not consider crossover in this paper as well.

Like the traditional mutation of GSGP, both IGSM and DGSM in SLIM adjust an individual's semantic coordinates by perturbing them of an amount contained within a specific range $[-\mathrm{MS}, \mathrm{MS}]$, where MS (mutation step) is a user-defined parameter. Geometric mutation shifts individuals within the solution space, with MS setting the maximum shift amplitude and significantly influencing the optimization process. Figure 1 visually illustrates the effect of MS. As widely shown in the literature, the ideal MS value depends on the problem at hand and is typically determined through hyperparameter tuning, often requiring time-consuming trial and error [6, 7]. Despite its importance in guiding the evolution process, few studies have specifically explored optimal configuration of the MS parameter.

For instance, [13] introduced an adaptive MS method, demonstrating that the optimal MS for each GSM event can be computed deterministically using the Moore-Penrose pseudoinverse [9]. However, while effective on training data, this approach often leads to overfitting, limiting its generalizability to new data. Similarly [17] employs an optimal-step one-tree GSGP mutation operator. This operator determines the optimal step for each mutation event by calculating the value that minimizes the distance of the resulting tree from the optimum. Al-

though this approach did not systematically outperform the standard application of GSGP.

Currently, hyperparameter tuning remains the most widely used approach, even if it is resource-intensive and often fails to identify the true optimal value, leading to less-than-ideal results and inefficient use of resources without guaranteeing generalization.

Selecting an appropriate MS value is closely tied to the dataset's specific characteristics, particularly the distribution and range of its input and output values. This is crucial since GSM involves exploring the solution space, and its effectiveness depends on how well the choice aligns with the dataset's properties. This study investigates how the scale of a dataset—whether in its input features, output features, or both—affects the selection of an appropriate MS for SLIM. Finding a general rule to automatically determine the optimal MS, or at least a close approximation, would eliminate the laborious and time-consuming process of manual hyperparameter tuning.

The paper is structured as follow: Section 2 introduces GSGP and SLIM. Section 3 describes the adopted methodology. Section 4 describes the datasets (artificial and real-world) used for the experimental study, as well as the used experimental settings. Section 5 provides a detailed analysis of the obtained results. Finally, Section 6 summarizes the main contributions of the paper and provides directions for further research.

2 Background

This section introduces the algorithms used in this study. First, in Section 2.1, we present GSGP, with a particular focus on GSM. Then, in Section 2.2, we describe the recently introduced variant SLIM.

2.1 Geometric Semantic Genetic Programming

When we tackle supervised learning problems, GP individuals are typically computer programs that map inputs into outputs. Given a GP individual P and a set of inputs $X = \{x_1, x_2, \ldots, x_n\}$, the output vector $s_P = \{P(x_1), P(x_2), \ldots, P(x_n)\}$ is referred to as the *semantics* of the individual P [20]. Traditionally, GP uses operators like mutation and crossover, which manipulate the syntactic structure of individuals. For example, crossover swaps subprograms between individuals, while mutation replaces subprograms with random programs. Though these operations are easy to describe, their effects on the semantics can be unpredictable. For instance, small syntactic changes can lead to significant semantic differences, compromising locality and thus complicating the search for optimal solutions [23]. To address this, semantic awareness was integrated into GP [32]. Among other approaches, GSGP was introduced by [20] as a variant of GP that replaces traditional syntax-based operators with GSOs. These operators, although modifying the syntax of the individuals, induce known geometric properties in the semantic space and, for any supervised learning problem, they generate a unimodal error surface.

Given a parent function $T : \mathbb{R}^n \to \mathbb{R}$, *Geometric Semantic Mutation* (GSM) generates the function $T_M = T + \text{MS} \cdot (T_{R1} - T_{R2})$, where MS is the mutation step and T_{R1} and T_{R2} are random real functions whose output ranges in the interval $[0, 1]$. This is usually obtained wrapping two random expressions with a sigmoid function. GSM generates an individual contained in the hyper-sphere of radius MS centred in the semantics of the parent in the semantic space. Building on the success of GSGP, researchers have explored many variations in recent years [3, 5, 22, 25]. Its major drawback is that it generates offspring larger than their parents, leading to final solutions that are hard to interpret [20].

2.2 Semantic Learning algorithm based on Inflate and deflate Mutation

As previously mentioned, one major drawback of GSGP is that the standard GSM causes a rapid increase in the size of individuals throughout the evolutionary process, making the final solutions difficult to interpret. Several solutions have been proposed to address this issue, such as dynamic population approaches [12] and the integration of gradient descent techniques [24, 25]. However, most of the existing methods still depend on the GSO introduced by [20] or utilize new operators that continue to produce offspring larger than their parents [2]. Recently, [33] introduced a new GSM that maintains the same properties as the traditional semantic mutation (renamed Inflate Geometric Semantic Mutation, or IGSM), while also producing smaller individuals than their parents. This new operator, called Deflate Geometric Semantic Mutation (DGSM), is based on two ideas. The first idea is that the IGSM defintion from the previous section can be rewritten as follows:

$$IGSM\,(T) = T + \text{MS} \cdot (T_{R1} - T_{R2}) = T - \text{MS} \cdot (T_{R2} - T_{R1})\,,$$

given that, trivially, $(T_{R1} - T_{R2}) = -(T_{R2} - T_{R1})$. The second idea is that the two random programs, T_{R1} and T_{R2}, are interchangeable because they are independent random variables with the same probability distribution. Therefore, IGSM can also be rewritten as:

$$IGSM\,(T) = T - \text{MS} \cdot (T_{R1} - T_{R2})\,.$$

Now, consider an individual T to which the IGSM has been applied, say, three times, resulting in the new individual:

$$T + \text{MS} \cdot (T_{R1} - T_{R2}) + \text{MS} \cdot (T_{R3} - T_{R4}) + \text{MS} \cdot (T_{R5} - T_{R6})\,.$$

Given that the two definitions are equivalent, we can now apply IGSM again using subtraction, resulting in the individual:

$$T + \text{MS} \cdot (T_{R1} - T_{R2}) + \text{MS} \cdot (T_{R3} - T_{R4}) + \text{MS} \cdot (T_{R5} - T_{R6}) - \text{MS} \cdot (T_{R7} - T_{R8})\,.$$

Clearly, the size of the individual continues to grow with each application of IGSM. However, if we apply IGSM with subtraction and reuse a pair of random programs from a previous mutation event such as, for instance, T_{R3} and

T_{R4}[3], we can simplify the expression to:

$$T + \text{MS} \cdot (T_{R1} - T_{R2}) \cancel{+\text{MS} \cdot (T_{R3} - T_{R4})} + \text{MS} \cdot (T_{R5} - T_{R6}) \cancel{-\text{MS} \cdot (T_{R3} - T_{R4})}.$$

The new individual has now a smaller genotype than its parent. This idea inspires the DGSM, which consists in the simple removal of a previously added term, and , as shown, is simply an alternative way of applying GSM by using subtraction and a previously used pair of random programs. DGSM retains the same semantic properties as GSM, perturbing each semantic coordinate by $[-\text{MS}, \text{MS}]$, and inducing a unimodal error surface for all supervised learning problems.

As the name SLIM suggests, this algorithm uses both IGSM and DGSM, as two independent operators, each with its own probability of being applied, with the only restriction that whenever a program contains only one term, only IGSM can be applied. More specifically, the number of times each GSM is applied is controlled by a parameter, which regulates the probability of applying DGSM instead of IGSM. When this parameter is set to zero, only IGSM is applied, making SLIM completely identical to traditional GSGP.

2.3 Different SLIM variants

Several variants of SLIM have been introduced, corresponding to as many ways to define the GSM operators. In this work, we consider three variants of SLIM, each using a different definition of GSM.

In the standard GSM defined in Section 2.1, a function with codomain $[-\text{MS}, \text{MS}]$ is *added* to the parent tree. Following [33], we refer to this mutation as +2SIG, since it involves two random trees, both wrapped in a sigmoid function. Recent studies [2, 30, 33] propose new ways to introduce ball mutations with range $[-\text{MS}, \text{MS}]$. Among the different variants, we focus on those that performed best in [33]. These are called *1SIG (representing one random expression wrapped in a sigmoid) and *ABS (representing absolute value).

Given a parent function $T : \mathbb{R}^n \to \mathbb{R}$, the *1SIG mutation with radius MS returns the function: $GSM\,(T) = T \cdot (1 + (\text{MS} \cdot (2 \cdot T_R - 1)))$, where T_R is a random function with codomain $[0, 1]$.

For the second variant, given a parent function $T : \mathbb{R}^n \to \mathbb{R}$, the *ABS mutation with radius MS returns the function: $GSM\,(T) = T \cdot \left(1 + \left(\text{MS} \cdot \left(1 - \frac{2}{1+|T_R|}\right)\right)\right)$, where T_R is a random function.

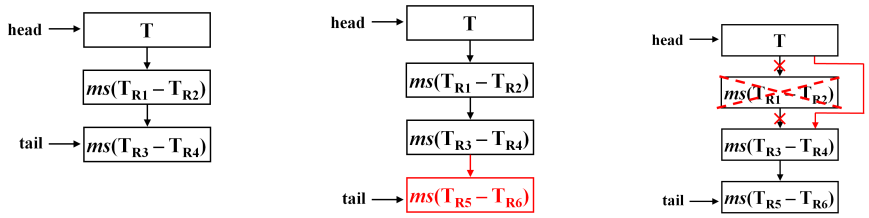
While the traditional GSM achieves a ball mutation by adding a positive or negative quantity centered at 0, the *1SIG and *ABS mutations achieve the same effect by multiplying by a quantity centered at 1. Unlike the +2SIG mutation, which uses two random functions, both *1SIG and *ABS use only one random function. As shown in [33], this modification reduces the growth rate of individuals during evolution while maintaining comparable performance to the standard GSM.

[3] Reusing random material is not new; it was done, for example, in [31].

These two new mutation definitions can be applied in both GSGP and SLIM, creating two new variants of these algorithms. In this work, we use the varinats that were proven to be have the best overall performance in [33]. Specifically we use the SLIM variants that employ the +2SIG, *1SIG, and *ABS GSM, denoted as SLIM+2SIG, SLIM*1SIG, and SLIM*ABS, respectively.

2.4 Linked-List Implementation of SLIM

As presented by [33], SLIM can be implemented by representing the genotype of each individual as a linked list of subexpressions. The IGSM appends one element at the end of the list, while the DGSM removes one element from the list. According to the variant used, the semantics of the individual can be obtained by accumulating the semantics of all the blocks using the sum (for SLIM+2SIG) or the product (for SLIM*1SIG and SLIM*ABS). A visual representation of the linked list implementation and the application of the IGSM and DGSM is presented in Figure 2.



(a) (b) (c)

Fig. 2: An example genotype of a SLIM individual, with a visual representation of the effect of the IGSM and DGSM on its genotype. When the IGSM is applied, a block is appended to the genotype of the individual, while when the DGSM is used, a block is removed.

3 Experimental Methodology

This study investigates how the intrinsic characteristics of a dataset affect the choice of an appropriate MS for GSMs, including both IGSM and DGSM. While different datasets require different MS values for effective evolution, this need likely depends on variation in input and/or output ranges or data distribution. For example, even if data may cover a large domain, if the majority of values are concentrated in a small subrange, this could influence the effectiveness of the MS more than the range of the domain itself. To determine which factors (input/output range and/or data distribution) are more influential, and based on that to develop a general rule to set the optimal MS, we present an experimental study consisting of two phases.

The first phase focuses on synthetic datasets, where we can control the input and output ranges. We generate different versions with varied ranges and test multiple MSs on them. These synthetic functions are designed to cover a wide range of complexities (non-linearity, multimodality, etc.) that commonly occur

in real-world datasets. Comparing these results allows us to derive a general rule, which will later be tested on real-world datasets where ranges cannot be controlled. In this phase, we do not impose any specific structure on the input distribution; instead, we generate random input values from a uniform distribution and compute target values using corresponding functions. After generating the datasets, we scale the input and output features using the following formula to map feature x to the desired $[a, b]$ range:

$$x_{\text{scaled}} = a + \left((b - a) \cdot \frac{(x - x_{\min})}{x_{\max} - x_{\min}} \right)$$

where $x_{\min}$ and $x_{\max}$ are the minimum and maximum values of x. We vary both the input range $[a_{in}, b_{in}]$ and the output range $[a_{out}, b_{out}]$, and create datasets with small-scale inputs and large-scale outputs, large-scale inputs and small-scale outputs, and cases where both input and output are either on a small or large scale. We then evolve SLIM using different MSs values, then rank performance to identify which dataset characteristics influence optimal MS selection and derive a general rule. More specifically, we will use synthetic datasets consisting of 1000 rows and 10 input columns, where the input and output values are constrained to a range of $[0, \max]$, with max selected from $1, 10, 100, 1000$. For each combination, we evaluated five MSs, corresponding to the ranges $[0, 0.1]$, $[0, 1]$, $[0, 10]$, $[0, 100]$, and $[0, 1000]$. At each mutation event, the mutation step is randomly drawn, with uniform probability, from this range.

The second phase of our study validates this hypothesis using real-world benchmark datasets. We analyze the distribution of input and output features and then evolve SLIM with different MSs to see if the best-performing ones match the rule inferred in the first phase. Unlike synthetic datasets, real-world data distributions are typically not uniform, so we need to determine whether the shape of the distribution affects the performance of different MSs. To address this, we test the MSs from the first phase, while also introducing new ones based on the statistical properties of the data distributions, such as the median, minimum, and maximum values. The objective of this second phase is to confirm or refine the rule established in the first phase and ultimately provide a definitive optimal value for the MS, that can be determined a priori by analyzing the data distribution, enabling an informed choice before training.

In summary, this work seeks to answer the following Research Questions (RQs):

- **RQ1:** Is the optimal MS influenced by any dataset feature? If so, is it affected by the output range, input range, or both?
- **RQ2:** Does the data distribution also influence the MS, or is it solely dependent on the scale of the input and output values?
- **RQ3:** Can we define a general rule that ensures a good MS without the need for hyperparameter tuning?

4 Material and Experimental Setup

This section describes the employed datasets (both synthetic and real-world) (Section 4.1) and provides all the experimental settings (Section 4.2) to make

the experiments completely reproducible. The code used for this experiment is available in the slim library.

4.1 Test Problems

As synthetic benchmarks, we select a set of functions widely used in the literature. These functions represent a variety of mathematical challenges and are commonly used to test the robustness and effectiveness of optimization algorithms. Table 1 outlines the functions used in this study, including their corresponding references. The chosen functions cover a broad range of characteristics, from multimodality and non-linearity (e.g., Ackley and Rastrigin) to simpler, unimodal problems (e.g., Sphere), allowing us to assess the impact of different MSs values across different landscapes.

Function Name	Formula	Reference
Ackley	$f(x) = -20\exp\left(-0.2\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2}\right) - \exp\left(\frac{1}{d}\sum_{i=1}^{d}\cos(2\pi x_i)\right) + 20 + e$	[1]
Alpine1	$f(x) = \sum_{i=1}^{d} \lvert x_i \sin(x_i) + 0.1x_i \rvert$	[29]
Alpine2	$f(x) = \prod_{i=1}^{d} \sqrt{x_i}\sin(x_i)$	[11]
Michalewicz	$f(x) = -\sum_{i=1}^{d} \sin(x_i)\left(\sin\left(\frac{ix_i^2}{\pi}\right)\right)^{2m}$	[19]
Rastrigin	$f(x) = 10d + \sum_{i=1}^{d}\left(x_i^2 - 10\cos(2\pi x_i)\right)$	[27]
Rosenbrock	$f(x) = \sum_{i=1}^{d-1}\left[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\right]$	[28]
Sphere	$f(x) = \sum_{i=1}^{d} x_i^2$	[15]

Table 1: Benchmark functions used in the first phase of our experimental study.

For the second phase, we conduct experiments on a range of real-world datasets from various domains. These datasets have often been used as benchmarks for GP, and their characteristics have been extensively analyzed in the literature [4, 12, 14, 33]. Table 2 provides an overview of their key features, including the number of instances and variables. For a more detailed description of these datasets, the reader is referred to [18].

Dataset	Observations	Features	Input Range	Target Range	Target Median	Target Std
yatch	307	6	[-5, 5]	[0, 62]	3	15.1
airfoil	1502	5	[0, 2000]	[103, 141]	125.7	6.9
slump	102	9	[0, 1050]	[0, 30]	21.5	8.7
strength	1029	8	[0, 1145]	[2.3, 83]	33.8	16.3
ppb	131	626	[-999, 2016472]	[0.5, 100]	85	31.1
bioav	359	241	[-71, 62651]	[0.4, 100]	75	30.5
ld50	234	626	[-999, 2289926]	[0.25, 89000]	775	2023.8

Table 2: Dataset specifications including the number of observations, features, input range, and target feature statistics.

4.2 Experimental Settings

Table 3 provides a detailed overview of the parameters utilized for the SLIM algorithm. Following [30], the probability of utilizing IGSM was set to 0.3 for all test datasets, with two exceptions: the LD50 dataset, where the probability

Parameter	Value
Generations	1000
Population Size	100
Elitism	True
Maximum Initial Depth	6
Function Set	$\{+, -, \times, \div\}$
Selection Algorithm	Tournament
Tournament Size	2
Initialization Method	Ramped Half-and-Half

Table 3: Parameter values used in the evolutionary algorithm.

was set to 0.1, and the concrete strength dataset, where it was increased to 0.5. The probability of utilizing DGSM was always set to 1 minus the probability of utilizing IGSM. At each application of the GSM, the mutation step is always generated from a uniform distribution in range $[0, \text{MS}]$, where MS is the parameter studied in this work. The investigation of the most appropriate value for MS is performed for all the three distinct SLIM versions (defined in Section 2), to determine whether different mutation definitions lead to varying optimal MS values. Specifically, the variants considered were SLIM+2SIG, SLIM*1SIG, and SLIM*ABS. The first, SLIM+2SIG, was selected as it is based on the standard mutation definition, while the others were included based on their excellent performance demonstrated in previous studies [30, 33].

We conducted 10 runs for each SLIM algorithm and each artificial dataset in the first phase, and 30 runs in the second phase. The Monte Carlo cross-validation [10, 34] was applied with each train-test partition randomly drawn at each different run, and with 70% of the observations forming the training set and the remaining 30% the test set. The partitions were consistent across all runs. We evaluated the results by computing the median of the Root Mean Squared Error (RMSE) across all runs.

5 Experimental Results

This section presents the results of the experiments described in Section 3. The goal is to analyze these results and determine how the range of the values in the dataset impacts the choice of an appropriate MS value, starting by inferring a general rule using artificial datasets (in Section 5.1) and then confirming the validity of our hypothesis with more complex real-world datasets (in Section 5.2). All the presented results are relative to the best individual in the population on the training set at each generation.

5.1 Experiments on the Artificial Datasets

Table 4 and Table 5 show the results from the first phase of the experimental study. RQ1 investigates whether input, output range, or both influence the

	In & Out Range Equal						Fix In Range						Fix Out Range					
***1SIG**		**Mutation Step**					***1SIG**		**Mutation Step**				***1SIG**		**Mutation Step**			
In	**Out**	0.1	1	10	10^2	10^3	**In**	**Out**	0.1	1	10	10^2	10^3	**In**	**Out**	0.1	1	10	10^2	10^3
1	1	**1**	**2**	**3**	**4**	**5**	1	1	**1**	**2**	**3**	**4**	**5**	1	1	**1**	**2**	**3**	**4**	**5**
10	10	**1**	**2**	**2**	**3**	**5**	1	10	**2**	**3**	**2**	**4**	**5**	10	1	**3.5**	**1**	**2**	**4**	**4.5**
10^2	10^2	**4**	**5**	**2**	**3**	**2**	1	10^2	**5**	**4**	**2**	**1**	**3**	10^2	1	**3.5**	**1**	**2**	**4**	**4**
10^3	10^3	**4**	**5**	**3**	**1**	**3**	1	10^3	**4**	**5**	**3**	**2**	**1**	10^3	1	**3**	**1**	**3**	**4**	**5**
***ABS**		**Mutation Step**					***ABS**		**Mutation Step**				***ABS**		**Mutation Step**			
In	**Out**	0.1	1	10	10^2	10^3	**In**	**Out**	0.1	1	10	10^2	10^3	**In**	**Out**	0.1	1	10	10^2	10^3
1	1	**1**	**2**	**3**	**4**	**5**	1	1	**1**	**2**	**3**	**4**	**5**	1	1	**1**	**2**	**3**	**4**	**5**
10	10	**1.5**	**2**	**2**	**3.5**	**5**	1	10	**1**	**2**	**3**	**4**	**5**	10	1	**3.25**	**1**	**2.5**	**4**	**4**
10^2	10^2	**4**	**5**	**2**	**2**	**2.5**	1	10^2	**5**	**3**	**2**	**1**	**4**	10^2	1	**3.5**	**1**	**2**	**4**	**4**
10^3	10^3	**4**	**4**	**3**	**1**	**2**	1	10^3	**5**	**3.5**	**3.5**	**2**	**1**	10^3	1	**3**	**1**	**2**	**4**	**5**
+2SIG		**Mutation Step**					**+2SIG**		**Mutation Step**				**+2SIG**		**Mutation Step**			
In	**Out**	0.1	1	10	10^2	10^3	**In**	**Out**	0.1	1	10	10^2	10^3	**In**	**Out**	0.1	1	10	10^2	10^3
1	1	**1**	**2**	**3**	**4**	**5**	1	1	**1**	**2**	**3**	**4**	**5**	1	1	**1**	**2**	**3**	**4**	**5**
10	10	**2**	**2**	**2**	**4**	**5**	1	10	**2**	**2**	**2**	**4**	**5**	10	1	**3**	**1**	**3**	**4**	**4**
10^2	10^2	**4**	**4**	**2**	**2**	**3**	1	10^2	**5**	**4**	**2**	**1**	**3**	10^2	1	**3**	**1**	**2**	**4**	**5**
10^3	10^3	**4**	**4**	**3**	**1**	**2**	1	10^3	**5**	**4**	**3**	**2**	**1**	10^3	1	**2**	**1**	**3**	**4**	**5**

Table 4: The results, as median rankings across all the datasets, of the first three experiments for all the studied algorithms.

	Exp 1						Exp 2						Exp 3					
***1SIG**		**Mutation Step**					***1SIG**		**Mutation Step**				***1SIG**		**Mutation Step**			
In	**Out**	0.1	1	10	10^2	10^3	**In**	**Out**	0.1	1	10	10^2	10^3	**In**	**Out**	0.1	1	10	10^2	10^3
10^2	1	**3.5**	**1**	**2**	**4**	**4**	10^3	1	**3**	**1**	**3**	**4**	**5**	1	1001	**5**	**4**	**3**	**2**	**1**
10^2	10	**1**	**2**	**3**	**3**	**5**	10^3	10	**1**	**2**	**3**	**4**	**5**	1	1010	**5**	**4**	**3**	**2**	**1**
10^2	10^2	**4**	**5**	**2**	**3**	**2**	10^3	10^2	**5**	**2**	**2**	**1**	**3**	1	1100	**5**	**4**	**3**	**1**	**2**
10^2	10^3	**2**	**4**	**5**	**3**	**2**	10^3	10^3	**4**	**5**	**3**	**1**	**3**	1	2000	**5**	**4**	**3**	**1**	**2**
***ABS**		**Mutation Step**					***ABS**		**Mutation Step**				***ABS**		**Mutation Step**			
In	**Out**	0.1	1	10	10^2	10^3	**In**	**Out**	0.1	1	10	10^2	10^3	**In**	**Out**	0.1	1	10	10^2	10^3
10^2	1	**3.5**	**1**	**2**	**4**	**4**	10^3	1	**3**	**1**	**2**	**4**	**5**	1	1001	**4**	**3.5**	**4**	**2**	**1**
10^2	10	**1**	**2**	**3**	**3**	**5**	10^3	10	**1**	**2**	**3**	**3**	**5**	1	1010	**5**	**3.5**	**3.5**	**2**	**1**
10^2	10^2	**4**	**5**	**2**	**2**	**2.5**	10^3	10^2	**5**	**3**	**2**	**2**	**3**	1	1100	**5**	**3.5**	**3.5**	**1**	**2**
10^2	10^3	**2**	**4**	**5**	**3**	**1.5**	10^3	10^3	**4**	**5**	**3**	**1**	**2**	1	2000	**5**	**3**	**4**	**1**	**2**
+2SIG		**Mutation Step**					**+2SIG**		**Mutation Step**				**+2SIG**		**Mutation Step**			
In	**Out**	0.1	1	10	10^2	10^3	**In**	**Out**	0.1	1	10	10^2	10^3	**In**	**Out**	0.1	1	10	10^2	10^3
10^2	1	**3**	**1**	**2**	**4**	**5**	10^3	1	**2**	**1**	**3**	**4**	**5**	1	1001	**5**	**4**	**3**	**2**	**1**
10^2	10	**2**	**2**	**3**	**3**	**5**	10^3	10	**1**	**2**	**3**	**4**	**5**	1	1010	**5**	**4**	**3**	**1**	**2**
10^2	10^2	**4**	**4**	**2**	**2**	**3**	10^3	10^2	**5**	**4**	**2**	**2**	**3**	1	1100	**5**	**4**	**3**	**1**	**2**
10^2	10^3	**3**	**4**	**5**	**3**	**2**	10^3	10^3	**4**	**4**	**3**	**1**	**2**	1	2000	**5**	**4**	**3**	**1**	**2**

Table 5: The results, as median rankings across all the datasets, of the last three experiments for all the studied algorithms.

optimal mutation step (MS). Table 4 explores this through three experiments: equal input-output ranges, a fixed input range with varying output, and a fixed output range with varying input. Variants of SLIM with different MSs were tested, and median rankings from best (1) to worst (5) are shown. For example, with SLIM*1SIG and a range of $[0, 1]$, the rankings for MS are: (1) $[0, 0.1]$, (2) $[0, 1]$, (3) $[0, 10]$, (4) $[0, 10^2]$, (5) $[0, 10^3]$. The left side of the table shows that, with equal input-output ranges, small ranges favor lower MSs, and larger ranges favor higher MSs, suggesting that the optimal MS typically matches or is slightly below the range's order of magnitude. This pattern holds across all algorithms tested. However, when only the output range varies (middle section of the table), rankings remain similar, implying that output range primarily drives MS performance. Fixing the output range at $[0, 1]$ (right side of the table) shows the best MS remains in the $[0, 1]$ range regardless of input range, reinforcing that input range does not affect the best MS selection.

This supports that MS operates within output space, where its magnitude must align with the output range. To further confirm, additional tests are presented in Table 5, which verify that optimal MSs match the output range's order of magnitude across varying input ranges. For example, with increased input ranges ($[0, 100]$ and $[0, 1000]$), optimal MSs are still determined by the output range alone. Finally, a test with the output range scaled by 1000 shows larger optimal MSs ($[0, 100]$, $[0, 1000]$), confirming that the MS depends on the output range scale rather than absolute values, emphasizing the importance of matching the order of magnitude.

5.2 Experiments on Real-World Datasets

The first phase of our study on synthetic datasets suggested that output scale largely determines the optimal mutation step (MS). Here, we test this on real-world datasets, which present additional challenges like noise and non-uniform distributions. Along with confirming our hypothesis for RQ1, we address RQ2 by examining whether the data distribution itself significantly influences MS selection. The results, shown in Figure 3, illustrate test fitness progression across generations for different SLIM versions (columns) and datasets (rows). We test MSs from Section 5.1 alongside two additional values based on output distribution: $[0, \text{median}]$ and $[\min, \max]$. As concluded from prior results, these values are based on the output range alone. The conclusions drawn from these plots are statistically validated by a Mann-Whitney U test (not shown here due to space limitations). Figure 4 presents the distributions of target variable for all the studied datasets.

Analyzing the results, the hypothesis that an optimal MS should match the order of magnitude of the output range holds on real-world datasets. When MS values are significantly lower than the target scale, models converge slowly to suboptimal solutions, as seen in the airfoil dataset (target range $[103, 141]$). Here, models with smaller MS values ($[0, 0.1]$, $[0, 1]$) yield higher test fitness, while larger values closer to the target scale ($[0, 100]$, $[0, 1000]$) show better performance. Conversely, a too-large MS relative to the output scale also hinders

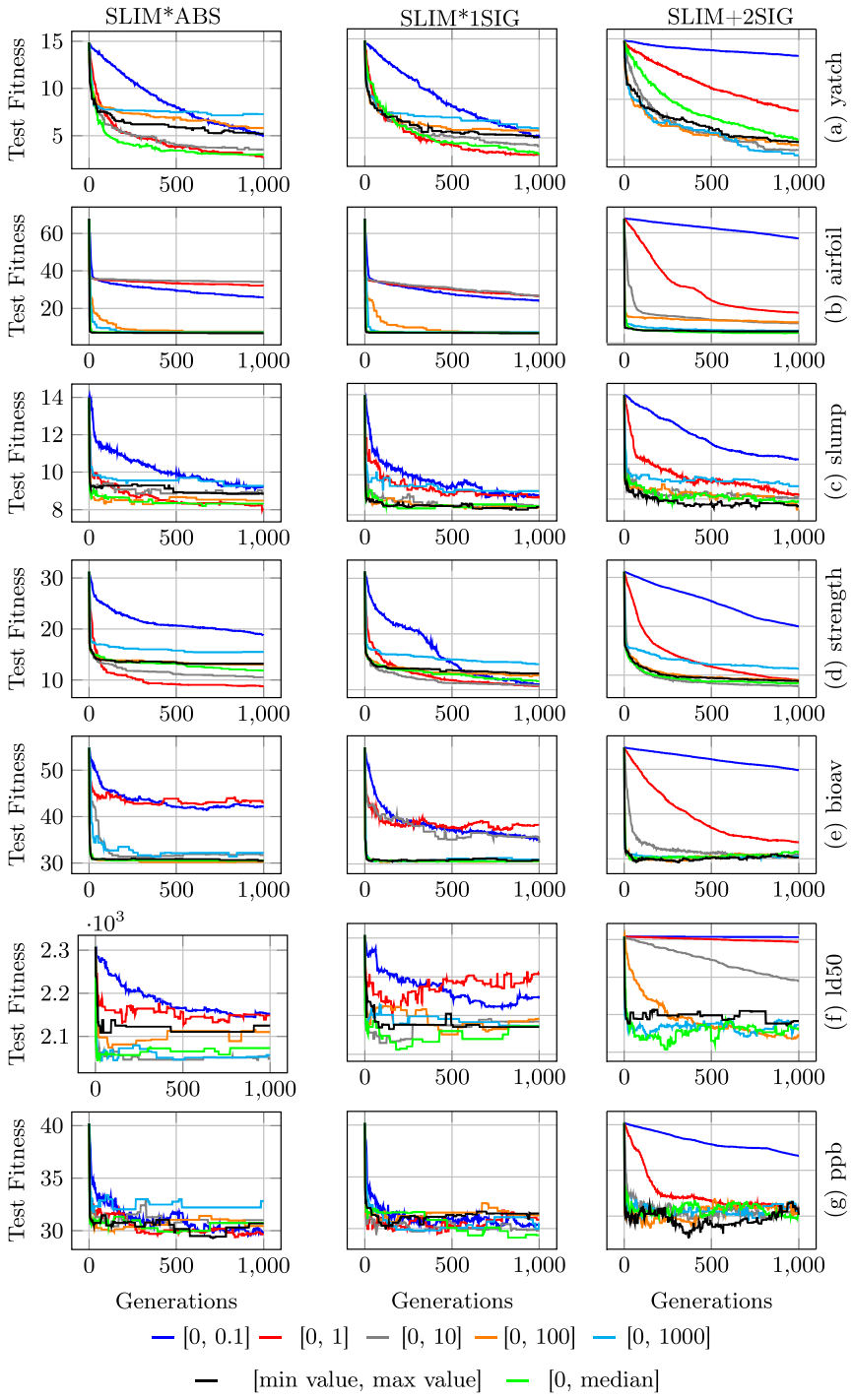


Fig. 3: Test Fitness of various datasets across three algorithms: SLIM*ABS, SLIM*1SIG, and SLIM+2SIG

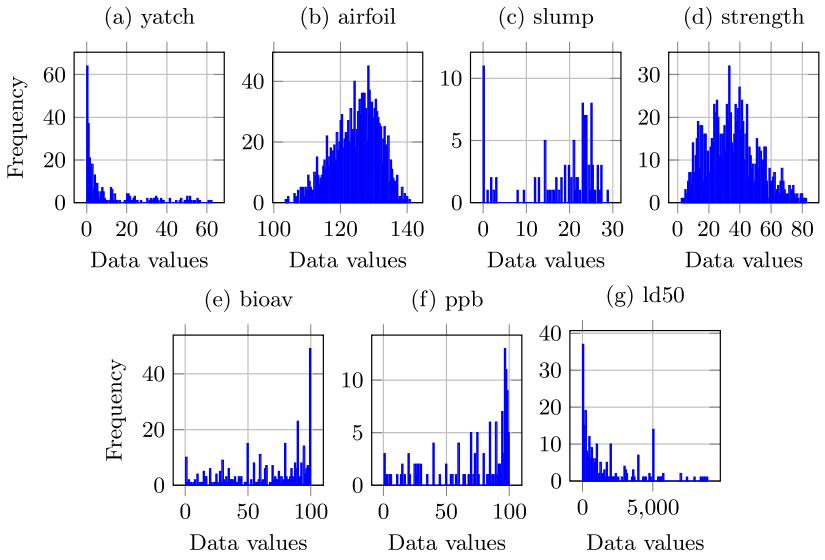


Fig. 4: Distribution of Target Variables for Various Datasets

convergence, as seen in the strength dataset (target range $[2.3, 80]$). A MS of $[0, 1000]$ leads to quick but suboptimal convergence. These patterns confirm that setting MS to align with the target scale supports optimal performance. However, real-world data reveal nuances beyond output range alone, particularly in datasets with skewed distributions. For instance, datasets like strength, bioav, and ppb have similar ranges, yet MS $[0, 1]$ performs well only in strength due to differences in data skewness. In the yacht dataset (target range $[0, 62]$), where values concentrate near the lower end, MS $[0, 1]$ performs well despite being smaller than the target range. These observations indicate that skewed distributions benefit from MSs that better reflect data density. The consistency of $[0, \text{median}]$ as a top-performing MS across datasets, as seen in Figure 3, underscores the importance of incorporating distribution properties. In contrast, the performance of $[\min, \max]$, which matches target scale but ignores data skew, varies with distribution. For instance, in the left-skewed ld50 dataset (range $[0, 89000]$, median 775), $[\min, \max]$ underperforms due to its disproportionate maximum. Thus, while $[\min, \max]$ performs well for symmetric or right-skewed distributions, left-skewed distributions benefit from MSs like $[0, \text{median}]$ that align with data density. In summary, both range and distribution shape inform MS selection. A MS smaller than the target range slows convergence to suboptimal solutions, while a too-large MS accelerates convergence but misses optimal solutions. For symmetric or right-skewed distributions, $[\min, \max]$ performs well,

whereas $[0, \text{median}]$ emerges as a robust choice across diverse data shapes, yielding optimal results across all datasets tested.

6 Conclusion and Future Work

This study presents an effective approach to optimizing the mutation step (MS) parameter in the Semantic Learning algorithm based on Inflate and deflate Mutation (SLIM) by basing its selection on dataset-specific characteristics. SLIM, a variant of Geometric Semantic Genetic Programming (GSGP), uses Inflate Geometric Semantic Mutation (IGSM) and Deflate Geometric Semantic Mutation (DGSM) to build compact, interpretable models, inducing a unimodal error surface. The MS parameter, crucial for SLIM's performance, traditionally requires extensive hyperparameter tuning to control offspring perturbation. Our findings streamline this process by establishing a practical selection rule for MS that eliminates manual tuning. Experiments on synthetic and real-world datasets showed that MS values aligning with the target output's magnitude, rather than input scale, improved convergence rate as well as accuracy. Real-world datasets confirmed that the target output scale is the main factor for effective MS selection. Skewed distributions performed best with MS centered on the target median, which improved model stability. Our selection rule based on $[0, \text{median}(\text{target})]$ offers a practical alternative to hyperparameter tuning, especially for complex datasets.

This work simplifies SLIM configuration, advancing its accessibility for machine learning and symbolic regression. Future research could investigate dynamic MS adjustments during the evolution or even the introduction of an individual-based MS, where each individual adapts MS differently from the others, based on its fitness, data distribution, diversity or other characteristics. Another future work scenario should focus on how feature standardization, as proposed in [8], affects the MS parameter choice. Finally, a Geometric Semantic Crossover (GSC) crossover, leveraging SLIM's linked-list structure, and generating individuals of a compact size, remains a valuable avenue for future investigation.

Acknowledgments

This work was supported by national funds through FCT (Fundação para a Ciência e a Tecnologia), under the project - UIDB/04152/2020 - Centro de Investigação em Gestão de Informação (MagIC)/NOVA IMS (https://doi.org/10.54499/UIDB/04152/2020).

References

[1] Ackley, D.H.: A connectionist machine for genetic hillclimbing (1987)

[2] Bakurov, I., Muñoz Contreras, J.M., Castelli, M., Rodrigues, N., Silva, S., Trujillo, L., Vanneschi, L.: Geometric semantic genetic programming with normalized and standardized random programs. Genetic Programming and Evolvable Machines **25**(1) (feb 2024). https://doi.org/10.1007/s10710-024-09479-1, https://doi.org/10.1007/s10710-024-09479-1

[3] Bonin, L., Rovito, L., De Lorenzo, A., Manzoni, L.: Cellular geometric semantic genetic programming. Genetic Programming and Evolvable Machines **25**(1), 8 (2024)

[4] Castelli, M., Manzoni, L., Gonçalves, I., Vanneschi, L., Trujillo, L., Silva, S.: An analysis of geometric semantic crossover: A computational geometry approach. pp. 201–208 (01 2016). https://doi.org/10.5220/0006056402010208

[5] Castelli, M., Trujillo, L., Vanneschi, L., Silva, S., Z-Flores, E., Legrand, P.: Geometric semantic genetic programming with local search. In: Proceedings of the 2015 annual conference on genetic and evolutionary computation. pp. 999–1006 (2015)

[6] Castelli, M., Vanneschi, L., Popovič, A.: Parameter evaluation of geometric semantic genetic programming in pharmacokinetics. International Journal of Bio-Inspired Computation **8**(1), 42–50 (2016). https://doi.org/10.1504/IJBIC.2016.074634, https://www.inderscienceonline.com/doi/abs/10.1504/IJBIC.2016.074634, pMID: 74634

[7] Dick, G.: An ensemble learning interpretation of geometric semantic genetic programming. Genetic Programming and Evolvable Machines **25**(1) (Mar 2024). https://doi.org/10.1007/s10710-024-09482-6, https://doi.org/10.1007/s10710-024-09482-6

[8] Dick, G., Owen, C.A., Whigham, P.A.: Feature standardisation and coefficient optimisation for effective symbolic regression. In: ACM Conferences, pp. 306–314. Association for Computing Machinery, New York, NY, USA (Jun 2020). https://doi.org/10.1145/3377930.3390237

[9] Dresden, A.: The fourteenth western meeting of the American Mathematical Society. Bulletin of the American Mathematical Society **26**(9), 385 – 396 (1920)

[10] Dubitzky, W., Granzow, M., Berrar, D.P.: Fundamentals of Data Mining in Genomics and Proteomics. Springer (2006)

[11] Epitropakis, M.G., et al.: Benchmark problems for cec 2013 special session on real-parameter optimization. In: Proceedings of the 2013 IEEE Congress on Evolutionary Computation. pp. 1096–1103 (2013)

[12] Farinati, D., Bakurov, I., Vanneschi, L.: A study of dynamic populations in geometric semantic genetic programming. Information Sciences **648**, 119513 (2023). https://doi.org/https://doi.org/10.1016/j.ins.2023.119513, https://www.sciencedirect.com/science/article/pii/S0020025523010988

[13] Gonçalves, I., Silva, S., Fonseca, C.M.: On the generalization ability of geometric semantic genetic programming. In: Machado, P., Heywood, M.I., McDermott, J., Castelli, M., García-Sánchez, P., Burelli, P., Risi, S., Sim, K. (eds.) Genetic Programming. pp. 41–52. Springer International Publishing, Cham (2015)

[14] Gonçalves, I., Silva, S., Fonseca, C.M., Castelli, M.: Unsure when to stop? In: Proceedings of the Genetic and Evolutionary Computation Conference. ACM (jul 2017). https://doi.org/10.1145/3071178.3071328

[15] Jong, K.D.: An analysis of the behavior of a class of genetic adaptive systems. Doctoral dissertation, University of Michigan (1975)

[16] Koza, J.R.: Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press, Cambridge, MA, USA (1992)

[17] McDermott, J., Agapitos, A., Brabazon, A., O'Neill, M.: Geometric Semantic Genetic Programming for Financial Data. In: Applications of Evolutionary Computation, pp. 215–226. Springer, Berlin, Germany (Nov 2014). https://doi.org/10.1007/978-3-662-45523-4_18

[18] McDermott, J., White, D.R., Luke, S., Manzoni, L., Castelli, M., Vanneschi, L., Jaśkowski, W., Krawiec, K., Harper, R., Jong, K.D., O'Reilly, U.M.: Genetic programming needs better benchmarks. In: Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation. pp. 791–798. GECCO '12, ACM, New York, NY, USA (2012). https://doi.org/10.1145/2330163.2330273

[19] Michalewicz, Z.: Genetic algorithms + data structures = evolution programs. Springer (1994)

[20] Moraglio, A., Krawiec, K., Johnson, C.G.: Geometric semantic genetic programming. In: Coello Coello, C.A., Cutello, V., Deb, K., Forrest, S., Nicosia, G., Pavone, M. (eds.) Parallel Problem Solving from Nature. LNCS, vol. 7491, pp. 21–31. Springer (2012)

[21] Moraglio, A., Mambrini, A.: Runtime analysis of mutation-based geometric semantic genetic programming for basis functions regression. In: *et al.*, E.A. (ed.) Proceedings of the annual international conference on Genetic and Evolutionary Computation. pp. 989–996. GECCO '13, ACM, New York, NY, USA (2013)

[22] Nadizar, G., Sakallioglu, B., Garrow, F., Silva, S., Vanneschi, L.: Geometric semantic gp with linear scaling: Darwinian versus lamarckian evolution. Genetic Programming and Evolvable Machines **25**(2), 1–24 (2024)

[23] Nguyen, Q.U.: Examining Semantic Diversity and Semantic Locality of Operators in Genetic Programming. Ph.D. thesis, University College Dublin, Ireland (18 Jul 2011), http://ncra.ucd.ie/papers/Thesis_Uy_Corrected.pdf

[24] Pietropolli, G., Manzoni, L., Paoletti, A., Castelli, M.: Combining geometric semantic gp with gradient-descent optimization. In: Medvet, E., Pappa, G., Xue, B. (eds.) Genetic Programming. pp. 19–33. Springer International Publishing, Cham (2022)

[25] Pietropolli, G., Manzoni, L., Paoletti, A., Castelli, M.: On the hybridization of geometric semantic gp with gradient-based optimizers. Genetic Programming and Evolvable Machines **24**(2), 16 (2023)
[26] Poli, R., Langdon, W.B., Mcphee, N.F.: A field guide to genetic programming (Mar 2008)
[27] Rastrigin, L.A.: Systems of extremal control. Moscow University (1974)
[28] Rosenbrock, H.H.: An automatic method for finding the greatest or least value of a function. The Computer Journal **3**, 175–184 (1960)
[29] Tang, K.S., et al.: Overlapping decomposition for scalability of GP-based function optimization. In: Proceedings of the 2007 IEEE Congress on Evolutionary Computation. pp. 2044–2051 (2007)
[30] Vanneschi, L.: SLIM_GSGP: The non-bloating geometric semantic genetic programming. In: Giacobini, M., Xue, B., Manzoni, L. (eds.) Genetic Programming. pp. 125–141. Springer Nature Switzerland, Cham (2024)
[31] Vanneschi, L., Castelli, M., Manzoni, L., Silva, S.: A new implementation of geometric semantic GP and its application to problems in pharmacokinetics. In: Krawiec, K., Moraglio, A., Hu, T., Uyar, A.S., Hu, B. (eds.) Proceedings of EuroGP. pp. 205–216. LNCS, Springer (2013)
[32] Vanneschi, L., Castelli, M., Silva, S.: A survey of semantic methods in genetic programming. Genetic Programming and Evolvable Machines **15**(2), 195–214 (Jun 2014). https://doi.org/10.1007/s10710-013-9210-0, https://doi.org/10.1007/s10710-013-9210-0
[33] Vanneschi, L., Farinati, D., Rasteiro, D., Rosenfeld, L., Pietropolli, G., Silva, S.: Exploring non-bloating geometric semantic genetic programming. Genetic Programming Theory and Practice (to appear)
[34] Vanneschi, L., Silva, S.: Lectures on Intelligent Systems. Natural Computing Series, Springer (2023). https://doi.org/10.1007/978-3-031-17922-8, https://doi.org/10.1007/978-3-031-17922-8
[35] Vanneschi, L., Silva, S., Castelli, M., Manzoni, L.: Geometric semantic genetic programming for real life applications. In: Riolo, R., Vladislavleva, K., Moore, J. (eds.) Genetic Programming Theory and Practice XI. Genetic and Evolutionary Computation, Springer US, Computer Science Collection (2013)