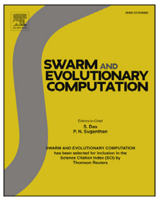
Swarm and Evolutionary Computation 44 (2019) 840–851

Contents lists available at ScienceDirect

Swarm and Evolutionary Computation

journal homepage: www.elsevier.com/locate/swevo



Alignment-based genetic programming for real life applications

Leonardo Vanneschi [a], Mauro Castelli [a,*], Kristen Scott [a], Leonardo Trujillo [b]

[a] *NOVA Information Management School (NOVA IMS), Universidade Nova de Lisboa, Campus de Campolide, 1070-312, Lisboa, Portugal*
[b] *Tree-Lab Instituto Tecnológico de Tijuana, Mesa de Otay, Tijuana, B.C., 22500, Mexico*

A R T I C L E I N F O

Keywords:
Genetic programming
Geometric semantic operators
Alignment
Error space
Real-life applications

A B S T R A C T

A recent discovery has attracted the attention of many researchers in the field of genetic programming: given individuals with particular characteristics of alignment in the error space, called optimally aligned, it is possible to reconstruct a globally optimal solution. Furthermore, recent preliminary experiments have shown that an indirect search consisting of looking for optimally aligned individuals can have benefits in terms of generalization ability compared to a direct search for optimal solutions. For this reason, defining genetic programming systems that look for optimally aligned individuals is becoming an ambitious and important objective. Nevertheless, the systems that have been introduced so far present important limitations that make them unusable in practice, particularly for complex real-life applications. In this paper, we overcome those limitations, and we present the first usable alignment-based genetic programming system, called nested alignment genetic programming (NAGP). The presented experimental results show that NAGP is able to outperform two of the most recognized state-of-the-art genetic programming systems on four complex real-life applications. The predictive models generated by NAGP are not only more effective than the ones produced by the other studied methods but also significantly smaller and thus more manageable and interpretable.

1. Introduction

The use of machine learning techniques to tackle challenging real-world problems is a cornerstone of the last decades. Among the various techniques inside the ML framework, evolutionary computation (EC) has shown its ability in addressing a plethora of complex problems over various domains [1]. The common idea shared by EC methods is to mimic the Darwinian principles of natural selection [2] to solve optimization problems. In particular, given an objective function that quantifies the quality of each solution, EC starts by randomly creating a set of candidate solutions and uses the objective function as an abstract fitness measure. Based on this fitness, some of the better candidates are chosen to seed the next generation by applying stochastic genetic operators [1].

One of the newest techniques belonging to the EC family is genetic programming (GP) [3], where solutions are (typically) represented as Lisp-like trees. GP was able to produce human-competitive results over various domains [4], and one of its advantages is that its application only requires a limited knowledge of the problem to be solved [3]. Despite its ability to produce good-quality solutions over various domains, GP suffers from some important limitations. The first issue relates to the time needed to evaluate each solution in the population, a process that is time-consuming and that can be exacerbated by the onset of bloat, the increase of tree size (i.e., the number of nodes) without a corresponding improvement in terms of fitness. The second limitation regards the use of the standard genetic operators used by GP: mutation and crossover. These two operators work by performing blind transformation of the syntax (i.e., the structure) of the solutions to build new individuals. While this allows for a simple definition of the genetic operators for various domains, the standard GP operators completely ignore the information about the behavior (i.e., the semantics) of the solutions. That is, given a parent solution (or two solutions for the crossover), it is very difficult to predict what will be the vector of output produced by the new child (two children for the crossover) after the application of genetic operators. Anyway, the semantics is what really matters for a GP practitioner and is the information necessary to evaluate the suitability of a solution in addressing a given problem. For this reason, the integration and use of semantic awareness in GP became one of the

* Corresponding author.
E-mail addresses: lvanneschi@novaims.unl.pt (L. Vanneschi), mcastelli@novaims.unl.pt (M. Castelli), m2015382@novaims.unl.pt (K. Scott), leonardo.trujillo@tectijuana.edu.mx (L. Trujillo).

https://doi.org/10.1016/j.swevo.2018.09.006
Received 11 January 2018; Received in revised form 18 September 2018; Accepted 20 September 2018
Available online 29 September 2018

hottest topics in the field of EC [5].

Semantics is defined as the output vector produced by a GP individual when evaluated over a set of training cases [5]. Among the many approaches that have been defined so far, geometric semantic GP (GSGP) [6] and one of its more recent developments, GSGP with local search (GSGP_LS) [7], have become particularly popular in the last few years. This success is probably due to the ability of these two systems to induce a unimodal error surface (i.e., with no local optima) for any supervised learning problem. Additionally, GSGP allows *direct* inclusion of semantic awareness in the evolutionary search process. On the other hand, the large majority of the existing semantic methods are indirect, in the sense that they use traditional syntax-based genetic operators and only accept newly created solutions based on some semantic criteria [5]. This feature makes GSGP suitable for addressing problems characterized by a vast amount of data, still maintaining an acceptable running time [8,9].

To continue this promising research stream, in this paper, we introduce a novel GP system, aimed at exploiting semantic awareness in a totally different way, compared to GSGP and GSGP_LS. The new system, called nested alignment GP (NAGP), is based on the recently defined concept of alignment in the error space, which is discussed in Section 2. The contribution of this work is twofold:

- In the first place, we deepen a very recent and promising idea to exploit semantic awareness in GP (i.e., the alignment in the error space [10]), which has received relatively little attention by the GP community so far;
- Secondly, we define a novel computational intelligence method, NAGP, which is able to generate predictive models that are more effective and manageable than the ones generated by GSGP and GSGP_LS for a set of complex real-life applications.

The former contribution is important for the GP community because it represents a further step forward in a very popular research line (improving GP with semantic awareness). On the other hand, considering that GSGP and GSGP_LS are regarded as state-of-the-art computational technology for generating predictive models in all the applications that we study in this paper, the latter contribution promises to have a tremendous impact on these very important applicative domains. As we will see in the rest of this paper, NAGP has two competitive advantages compared to GSGP and GSGP_LS: Not only is NAGP able to obtain more accurate predictive models, but these models are also smaller in size, which makes them more readable and interpretable.

The paper is organized as follows: In Section 2, we introduce the idea of alignment in the error space, also discussing some previous preliminary studies in which this idea was developed. In Section 3, we discuss the studied applications. In Section 4, we present NAGP for the first time, as well as a variant of NAGP called NAGP_β, describing every single step of their implementation. Section 5 describes the employed experimental settings. In Section 6, we present and discuss the obtained experimental results, comparing the performance of NAGP and NAGP_β to that of GSGP and GSGP_LS for the prediction of the energy consumption of buildings. Finally, Section 7 concludes the paper and proposes suggestions for future research.

2. Previous work on alignment in the error space

A few years after the introduction of GSGP, a new way of exploiting semantic awareness was presented in Ref. [10] and further developed in Refs. [11,12]. The idea, which is also the focus of this paper, is based on the concept of error space, which is illustrated in Fig. 1. In the genotypic space, programs are represented by their syntactic structures (for instance, trees, as in Ref. [3], or any other of the existing representations). As explained above, semantics can be represented as a point in a space that we call semantic space. In supervised learning, the target is also a point in the semantic space, but usually (except in the rare case where the target value is equal to zero for each training case), it does *not* correspond to the origin of the Cartesian system. Then, we translate each point in the semantic space by subtracting the target. In this way, for each individual, we obtain a new point, which we call an *error vector*, and we call the corresponding space the *error space*. The target, by construction, corresponds to the origin of the Cartesian system in the error space. In Ref. [10], the concepts of optimally aligned and optimally coplanar individuals were introduced, along with their important implications, which are summarized here.

Two individuals, A and B, are *optimally aligned* (or *optimally collinear*) if a scalar constant k exists, such that

$$\vec{e_A} = k \cdot \vec{e_B}, \tag{1}$$

where $\vec{e_A}$ and $\vec{e_B}$ are the error vectors of A and B respectively. Based on this definition, it is not difficult to see that two individuals are optimally aligned if the straight line joining their error vectors also intersects with the origin in the error space. This property is graphically shown in Fig. 2(a). Analogously, and extending the idea to three dimensions, three individuals are *optimally coplanar* if the bidimensional plane in which their error vectors lie in the error space also intersects with the origin. This property is shown in Fig. 2(b).

In Ref. [10], it is proven that given any pair of optimally aligned individuals A and B, it is possible to reconstruct a globally optimal solution P_{opt}. This solution is defined in Equation (2):

$$P_{opt} = \frac{1}{1-k} \cdot A - \frac{k}{1-k} \cdot B, \tag{2}$$

where k is the same constant as in Equation (1). This optimal solution is represented in a tree shape in Fig. 3. Analogously, in Ref. [10], it was also proven that given any triplet of optimally coplanar individuals, it is possible to analytically construct a globally optimal solution (refer to [10] for the equation of the globally optimal solution in that case). As Fig. 2(b) shows, the tridimensional property is just an iteration of the bidimensional one; in fact, if three individuals, A, B, and C, are optimally coplanar, it is always possible to find a vector $\vec{m}$ that is aligned with $\vec{e_A}$ and $\vec{e_B}$ and also aligned with $\vec{e_C}$ and the origin.

Several possible ways of searching for alignments can be imagined. In Ref. [10], one.

Preliminary attempt was made by fixing one direction, called the *attractor*, and "pushing" all the individuals in the population toward alignment with the attractor. In this way, it is possible to maintain the traditional representation of solutions, where each solution is represented by one program. The other face of the coin is that, in this way, we strongly restrict what GP can do, forcing the alignment to necessarily happen in just one prefixed direction (i.e., the direction of the attractor). One of the objectives of this paper is to relieve this constraint by defining a new GP system that is generally able to evolve *vectors* of programs (even though only vectors of size equal to 2 will be used in this paper).

In Ref. [11], a first attempt to use a multiple program representation was made. In that work, individuals were pairs of programs, and fitness was the angle between the respective error vectors. This situation is graphically represented in Fig. 4, where the two error vectors are called $\vec{a}$ and $\vec{b}$, and the angle used as fitness is called θ. It is obvious that if θ is equal to zero, then $\vec{a}$ and $\vec{b}$ are aligned with each other and with the origin. Thus, the objective of GP is to minimize θ. In this way, alignments could be found in any possible direction of the error space, with no restrictions. From now on, for the sake of clarity, this type of individual (i.e., individuals characterized by more than one program) will be called *multi-individuals*. In Ref. [11], the following problems of this approach were reported:

- generation of semantically identical, or very similar, expressions;
- k constant in Equation (2) equal, or very close, to zero;
- generation of expressions with huge error values.

These problems strongly limited the work, to the point that the approach itself was considered unusable in practice in Ref. [11]. These

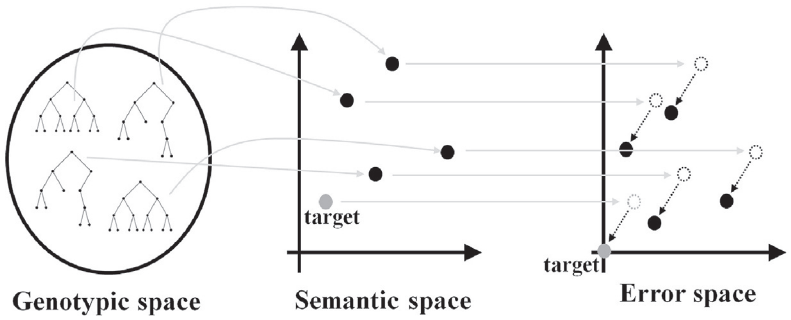


Fig. 1. Programs are represented by trees (or any other structures, such as linear genomes, graphs, etc.) in the genotypic space. Each one maps into its semantics, identified by a point in the semantic space. The semantics are then translated by subtracting the target, obtaining a point in the error space. The target, which usually does not correspond to the origin of the Cartesian system in the semantic space corresponds to the origin in the error space by construction.

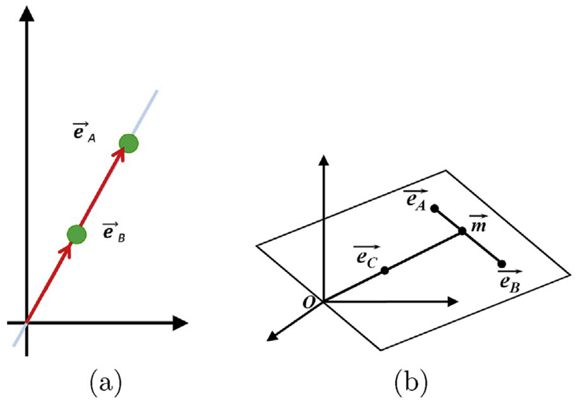


Fig. 2. A simple graphical representation of optimally aligned and optimally coplanar individuals. Plot (a): Two optimally aligned individuals, *A* and *B*. Plot (b): Three optimally coplanar individuals, *A*, *B*, and *C*.

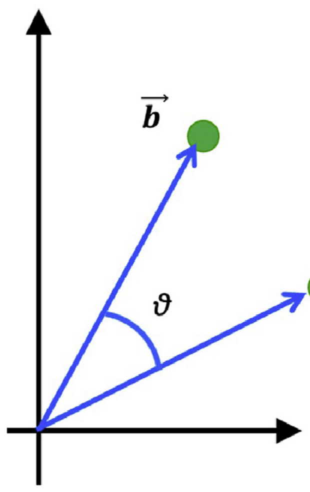


Fig. 4. A GP system can search for pairs of optimally aligned programs by evolving a population of individuals that are pairs of programs. In that case, theoretically, the fitness can simply be the angle θ between the error vectors of the two programs. However, as the text explains, this simple method has problems that need to be overcome. This is done by NAGP, the system presented in this paper.

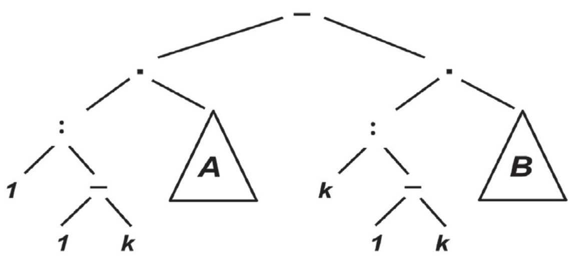


Fig. 3. A tree representation of the globally optimal solution P_{opt}, where *A* and *B* are optimally aligned programs.

problems are discussed here, and in Section 4 we describe how the proposed method, NAGP, overcomes them.

Issue 1: Generation of semantically identical, or very similar, expressions. A simple way for GP to find two expressions that are optimally aligned in the error space is to find two expressions that have exactly the same semantics (and consequently the same error vector). However, this causes a problem once we try to reconstruct the optimal solution, as in Equation (2). In fact, if the two expressions have the same error vector, the k value in Equation (2) is equal to 1, which gives a denominator equal to zero. In some preliminary experiments, we have seen that GP tends very often to generate multi-individuals that have this kind of problem. Also, it is worth pointing out that even if GP is prevented from generating multi-individuals that have identical semantics, GP may still push the evolution toward the generation of multi-individuals whose expressions have semantics that are very similar. This leads to a k constant in Equation (2) that, although not being exactly equal to 1, has a value that is very close to 1. As a consequence, the denominator in Equation (2), although not being exactly equal to zero, may be very close to zero, and thus, the value calculated by Equation (2) could be a huge number. This would force a GP system to deal with unbearably large numbers during all its execution, which may lead to several problems, including numeric overflow.

Issue 2: The *k* constant in Equation (2) **is equal, or very close, to zero.** Looking at Equation (2), one may notice that if k is equal to zero, then expression B is irrelevant and the reconstructed solution P_{opt} is equal to expression A. A similar problem also manifests itself when k is not exactly equal to zero but very close to zero. In this last case, both expressions A and B contribute to P_{opt}, but the contribution of B may be so small that is considered marginal, and P_{opt} would *de facto* be extremely similar to A. In some preliminary experiments, we have seen that unless this issue is taken care of, the evolution very often generates such situations. This basically turns a multi-individual alignment-based system into traditional GP, in which only one of the programs in the multi-individual matters. If we really want to study the effectiveness of multi-individual alignment-based systems, we have to impede this kind of situation.

Issue 3: Generation of expressions with huge error values. Theoretically speaking, systems based on the concept of alignment in the error space could limit themselves to searching for expressions that are optimally aligned, without taking into account their performance (i.e., how close their semantics are to the target). However, in some preliminary experiments we have seen that if we give GP the sole task of finding aligned expressions, GP frequently tends to generate expressions whose semantics contain unbearably large values. Once again, this may lead to several problems, including numeric overflow, and a successful system should definitely prevent this from happening.

One fact that should be mentioned is that none of the previous issues can be taken into account with simple conditions that prevent certain situations from happening. For instance, one may consider solving Issue 1 by simply testing if the expressions in a multi-individual are semantically identical to each other and rejecting the multi-individual if that happens. However, as already discussed, expressions that have very similar semantics may also lead to problems. Furthermore, the idea of introducing a threshold ϵ into the semantic diversity of the expressions in a multi-individual, and rejecting all the multi-individuals for which the diversity is smaller than ϵ, does not seem to be a brilliant solution. In fact, in some preliminary experiments, we have seen that GP tended to generate multi-individuals with a diversity equal, or very close, to ϵ itself. Analogously, if we consider Issue 2, neither rejecting multi-individuals that have a k constant equal to zero nor rejecting individuals that have an absolute value of k larger than a given threshold would solve the problem. Finally, considering Issue 3, rejecting individuals that have the coordinates of the semantic vector larger than a given threshold $\delta_{\max}$ would not solve the problem because GP would tend to generate expressions in which the coordinates of the semantic vector are equal, or very close, to $\delta_{\max}$ itself.

In such a situation, we believe that a promising way to effectively solve these issues (besides defining the specific conditions mentioned above) is to take the issues into account in the selection process, for instance, giving more probability of being selected for mating to multi-individuals that have large semantic diversity between the expressions, values of k that are, as much as possible, far from zero or one (which represents identical expressions) and expressions whose semantics are, as much as possible, close to the target. These ideas are implemented in NAGP, which is described in the following section. The definition of such a selection process extends a research area that is particularly relevant in EC. For instance, the work of Nalepa and Czech [13] summarized selection schemes applied in an evolutionary algorithm for discrete optimization problems, while in Ref. [14], the authors presented various selection schemes for artificial bee colony optimization.

3. Test problems and data sets

To test the studied GP frameworks, we used four benchmark problems. The first dataset was already used in Ref. [15] and in Ref. [16]. In particular, the dataset consists of 8 independent variables (or features) and 768 instances. Each instance is related to a particular building, and the objective is to predict the energy consumption of the heating system of a particular building. An explanation of the features is reported in Table 1. For additional details about the dataset, the reader is referred to [15]. The dataset is freely available at the UCI machine learning repository (http://archive.ics.uci.edu/ml/datasets/Energy+efficiency).

The second benchmark is related to the prediction of high-performance concrete strength. The dataset can be downloaded from the UCI machine learning repository (http://archive.ics.uci.edu/ml/datasets/Concrete+Compressive+Strength). The dataset was already used in [17], where the procedure adopted to collect the data was fully described and used in previous studies about GP [18]. The dataset consists of 1028 instances, each of them described by 8 variables that are reported in Table 2.

The last two benchmarks are related to the field of pharmacokinetics. In particular, the first application refers to human oral bioavailability, a parameter that measures the percentage of initial drug dose that effectively reaches the systemic blood circulation [19]. This parameter is particularly relevant for pharmaceutical industries because oral administration is usually the preferred way to supply drugs to patients

Table 1
Features and number of their possible values for the energy problem.

Features	Description	Number of possible values
X1	Relative compactness	12
X2	Surface area	12
X3	Wall area	7
X4	Roof area	4
X5	Overall height	2
X6	Orientation	4
X7	Glazing area	4
X8	Glazing area distribution	6
Y1	Target variable representing the heating load (HL)	586

and because it is a representative measure of the quantity of active principle that effectively can actuate its biological effect. The dataset consists of 241 variables and 360 instances. For a complete description of the dataset, the reader is referred to [20]. The second pharmacokinetics dataset refers to a parameter called protein plasm binding (PPB) level. This value corresponds to the percentage of the initial drug dose that binds plasma proteins. As reported in [19], this measure is fundamental, both because blood circulation is the major vehicle of drug distribution into the human body and because only free (unbound) drugs permeate the cellular membranes and reach the targets. The dataset consists of 626 variables and 234 instances.

4. Nested alignment genetic programming

NAGP uses multi-individuals and thus extends the first attempt proposed in Ref. [11]. In this section, we describe the selection, mutation, and population initialization of NAGP, keeping in mind that no crossover has been defined yet for this method. Furthermore, we explain how NAGP overcomes the problems described in Section 2. In the last part of this section, we also define a variant of the NAGP method, called NAGP_β, which will also be taken into account in our experimental study.

Selection Aside from trying to optimize the performance of the multi-individuals, selection is the phase that takes into account the issues described in Section 2. NAGP contains five selection criteria, which have been organized into a nested tournament. Let $\phi_1, \phi_2, \dots, \phi_m$ be the expressions characterizing a multi-individual. It is worth pointing out that only the case $m = 2$ is taken into account in this paper. However, the concept is general, so it is explained using m expressions. The selection criteria are:

- Criterion 1: diversity (calculated using the standard deviation) of the semantics of the expressions $\phi_1, \phi_2, \dots, \phi_m$ (to be maximized).
- Criterion 2: the absolute value of the k constant that characterizes the reconstructed expression P_{opt} in Equation (2) (to be maximized).
- Criterion 3: the sum of the errors of the single expressions $\phi_1, \phi_2, \dots, \phi_m$ (to be minimized).
- Criterion 4: the angle between the error vectors of the expressions $\phi_1, \phi_2, \dots, \phi_m$ (to be minimized).
- Criterion 5: the error of the reconstructed expression P_{opt} in Equation (2) (to be minimized).

The nested tournament works as follows: An individual is selected if it is the winner of a tournament, which we call T_5, based on Criterion 5. All the participants in tournament T_5, instead of being individuals chosen at random as in the traditional tournament selection algorithm, are winners of previous tournaments (that we call tournaments of type T_4), which are based on Criterion 4. Analogously, for all $i = 4, 3, 2$, all participants in the tournaments of type T_i are winners of previous tournaments (that we will call tournaments of type T_{i-1}), based on Criterion

Table 2
Features characterizing the concrete problem.

ID	Name (Unit Measure)	Minimum	Maximum	Mean	Median	Std. Dev.
$X0$	Cement (kg/m^3)	102	540	281.2	272.9	104.5
$X1$	Fly ash (kg/m^3)	0	359.4	73.9	22	86.3
$X2$	Blast furnace (kg/m^3)	0	200.1	54.2	0	64
$X3$	Water (kg/m^3)	121.8	247	181.6	185	21.4
$X4$	Superplasticizer (kg/m^3)	0	32.2	6.2	6.4	6
$X5$	Coarse aggregate (kg/m^3)	801	1145	972.9	968	77.8
$X6$	Fine aggregate (kg/m^3)	594	992.6	773.6	779.5	80.2
$X7$	Age of testing (days)	1	365	45.7	28	63.2

$i - 1$. Finally, the participants in the tournaments of type T_1 (the kind of tournament that is based on Criterion 1) are individuals selected at random from the population. In this way, an individual, in order to be selected, has to undergo five selection *layers* (following a lexicographic ordering of the criteria), each of which is based on one of the five different chosen criteria. A preliminary set of experiments has shown that the order in which we apply these criteria is not relevant. Motivations for the chosen criteria are as follows:

- Criterion 1 was introduced to counteract Issue 1 in Section 2. Maximizing the semantic diversity of the expressions in a multi-individual should naturally prevent GP from creating multi-individuals with identical semantics or semantics that are very similar.
- Criterion 2 was introduced to counteract Issue 2 in Section 2. Maximizing the absolute value of constant k should naturally allow GP to generate multi-individuals for which k's value is neither equal nor close to zero.
- Criterion 3 was introduced to counteract Issue 3 in Section 2. If the expressions that characterize a multi-individual have a "reasonable" error, then their semantics should be reasonably similar to the target, thus naturally avoiding the appearance of unbearably large numbers.
- Criterion 4 is a performance criterion: If the angle between the error vectors of the expressions $\phi_1, \phi_2, \ldots, \phi_m$ is equal to zero, then Equation (2) allows us to reconstruct a perfect solution P_{opt} (see Fig. 4 for the bidimensional case). Also, the smaller this angle, the smaller the error of P_{opt} should be. Nevertheless, in some preliminary experiments, we have noticed that multi-individuals may exist with similar values of this angle, but very different values of the error of the reconstructed solution P_{opt}, due, for example, to individuals with a very large distance from the target. This fact made us conclude that Criterion 4 cannot be the only performance objective and suggested that we also introduce Criterion 5.
- Criterion 5 is a further performance criterion. Among multi-individuals with the same angle between the error vectors of the expressions $\phi_1, \phi_2, \ldots, \phi_m$, the preferred ones will be the ones for which the reconstructed solution P_{opt} has the smallest error.

Mutation The mechanism we have implemented for applying mutation to a multi-individual is extremely simple: For each expression ϕ_i in a multi-individual, mutation is applied to ϕ_i with a given mutation probability p_m, where p_m is a parameter of the system. It is worth remarking that in our implementation all expressions ϕ_i of a multi-individual have the same probability of undergoing mutation, but this probability is applied independently to each one. Therefore, some expressions could be mutated, and others could remain unchanged. The type of mutation that is applied to expressions is Koza's standard subtree mutation [3].

To this "basic" mutation algorithm, we have also decided to add a mechanism of rejection in order to help the selection process counteract the issues discussed in Section 2. Given a prefixed parameter that we call δ_k, if the multi-individual generated by mutation has a k constant included in the range $[1 - \delta_k, 1 + \delta_k]$, or in the range $[-\delta_k, \delta_k]$, then the k constant is considered, respectively, too close to 1 or too close to 0 and the multi-individual is rejected. In this case, a new individual is selected for mutation, again using the nested tournament discussed above.

The combined effect of this rejection process and of the selection algorithm should strongly counteract the issues discussed in Section 2. In fact, when k is equal to 1, or equal to 0, or even close to 1 or 0 inside a given prefixed toleration radius δ_k, the multi-individual is not allowed to survive. For all the other multi-individuals, the distances between k and 1 and between k and 0 are used as optimization objectives to be maximized. This allows NAGP to evolve multi-individuals with k values that are "reasonably far" from 0 to 1.

Initialization NAGP initializes a population of multi-individuals using multiple executions of the ramped half and half algorithm [3]. More specifically, let n be the number of expressions in a multi-individual ($n = 2$ in our experiments), and let m be the size of the population that has to be initialized. NAGP runs the ramped half and half algorithm n times, thus creating n "traditional" populations of programs P_1, P_2, …, P_n, each of which contains m trees. Let $P = \{\Pi_1, \Pi_2, \ldots, \Pi_m\}$ be the population that NAGP has to initialize (where for each $i = 1, 2, \ldots, m$, Π_i is an n-dimensional multi-individual). Then, for each $i = 1, 2, \ldots, m$ and for each $j = 1, 2, \ldots, n$, the jth program of multi-individual Π_i is the jth tree in population P_i.

To this "basic" initialization algorithm, we have added an adjustment mechanism to make sure that the initial population does not contain multi-individuals with a k equal, or close, to 0 or 1. More specifically, given a prefixed number of expressions α, which is a new parameter of the system if the created multi-individual has a k value included in the range $[1 - \delta_k, 1 + \delta_k]$, or in the range $[-\delta_k, \delta_k]$ (where δ_k is the same parameter as the one used for implementing rejections of mutated individuals), then α randomly chosen expressions in the multi-individual are removed and replaced by the same number of new randomly generated expressions. Then, the k value is calculated again, and the process is repeated until the multi-individual has a k value that stays outside the ranges $[1 - \delta_k, 1 + \delta_k]$ and $[-\delta_k, \delta_k]$. Only when this happens, is the multi-individual accepted inside the population. Given that only multi-individuals of two expressions are considered in this paper, in our experiments we always used $\alpha = 1$.

Aside from NAGP, the following variant was also implemented:

NAGP_β. This method integrates a multi-individual approach with a traditional single-expression GP approach. More precisely, the method begins as NAGP, but after β generations, the evolution is done by GSGP. In order to "transform" a population of multi-individuals into a population of traditional single-expression individuals, each multi-individual is replaced by the reconstructed solution P_{opt} in Equation (2). The rationale behind the introduction of NAGP_β is that alignment-based systems are known to have a very quick improvement in fitness in the first generations, which may sometimes cause overfitting of training data (refer to [10–12] for a discussion of the issue). Given that GSGP is instead known for being a slow optimization process, able to limit overfitting under certain circumstances (see Ref. [21]), the idea is to transform NAGP into GSGP, possibly before overfitting arises. Even though a deep study of parameter β is strongly in demand, only $\beta = 50$ was tested in this paper. For this reason, from now on, the name NAGP_50 will be

Table 3
GP parameters used in our experiments.

Parameter	Setting
Population size	100
Max. numb. of generations	200
Initialization	Ramped H-H
Maximum depth for evolution	17
Maximum depth for initialization	6
δ_k	0.02

used for this method.

5. Experimental settings

Performance of the evaluated systems was assessed by considering a *k*-fold cross-validation, in order to avoid any bias with respect to the splitting of the data. In particular, a repeated 10-fold cross-validation (for a total of 30 runs) was executed to ensure statistical robustness of the results. The fitness function is the root mean square error (RMSE) between target and obtained values. The parameters used are summarized in Table 3. These values were obtained after a preliminary tuning phase (performed by adopting a grid search procedure), and are the values that have allowed us to obtain the best results. Grid search is one of the commonly used methods for tuning ECs' parameters, and additional information can be found in the work of Zhang and coauthors [22], where various approaches for automating the tuning process were discussed. Aside from those parameters, the primitive operators were addition, subtraction, multiplication, and division protected as in Ref. [3]. The terminal symbols included one variable for each feature in the dataset, plus the following numerical constants: −1.0, −0.75, −0.5, −0.25, 0.25, 0.5, 0.75, 1.0. Parent selection was done using tournaments of size 2 for GSGP and GSGP_LS, and tournaments of the same size for each layer of the nested selection for NAGP and in the first 50 generations of NAGP_50. Crossover rate was equal to zero (i.e., no crossover was performed during the evolution) for all the studied methods. While NAGP and NAGP_50 do not have a crossover operator implemented yet, the motivations for not using crossover in GSGP and GSGP_LS can be found in Ref. [11] and are related to the geometric properties of the semantic operators.

6. Experimental results

In this section, we present the results of the experimental campaign. In particular, results are discussed by first comparing the performance of the different GP based systems and, subsequently, by comparing the performance of NAGP and NAGP_50 against that produced by various machine learning techniques that are commonly used to address regression problems.

Fig. 5 reports the training and test fitness achieved by the various GP systems on the four benchmarks taken into account. In particular, the left-side boxplots report the performance on the training set, while the right-side boxplots contain the fitness achieved over unseen instances. The evolutionary plots of the same runs are displayed in Fig. 6, where median values obtained over all the considered runs are reported. The discussion starts with the analysis of the training fitness. As one can see from Fig. 5(a), for the bioavailability dataset, the best performer is GSGP, followed by NAGP_50, GSGP_LS, and NAGP. A completely different behavior appears in Fig. 5(c), for the concrete dataset. In this case, NAGP_50 is the best performer, followed by GSGP_LS and NAGP. Hence, focusing on the two proposed variants, it seems that the idea of combining the nested tournament selection and a traditional GSGP search process can be beneficial for achieving satisfactory performance. In the concrete dataset, the worst performer is GSGP, while GSGP coupled with a local search strategy produces a comparable performance with respect to NAGP. The difference in terms of performance between NAGP and NAGP_50 is even clearer considering the energy dataset (Fig. 5(e)). In this case, NAGP_50 is the best performer, followed by GSGP and GSGP_LS (which produce comparable results). On the other hand, NAGP is the worst performer, hence corroborating the idea that combining the tournament selection with a standard GSGP search process is beneficial for achieving good-quality solutions. We believe that this feature is related to the ability of NAGP_50 to explore the search space without the constraints of NAGP in terms of nested tournament selection. In other words, NAGP_50 makes use of the nested tournament selection to find good-quality solutions and, once NAGP starts to converge, gives GSGP the chance to eventually find better quality solutions. This trend is also corroborated by the results displayed in Fig. 5(g), where NAGP_50 is the best performer, followed by GSGP_LS and GSGP. Also in this case, NAGP is the worst performer, thus strengthening the hypothesis that using the nested tournament selection excessively limits the ability of GP to explore various areas of the search space. On the other hand, the use of GSGP after NAGP is beneficial for improving the performance of the solutions found by the latter search process.

To perform a statistical validation of the results obtained on the training instances, we performed various tests. In particular, for each technique, we considered the fitness (RMSE) of the best individual at the last generation, thus resulting in a series of 30 values. The values are the same as those used to build the boxplots displayed in Fig. 5. The Lilliefors test showed that the data are not normally distributed and hence, a rank-based statistic was used. The Wilcoxon rank-sum test for pairwise data comparison with Bonferroni correction was used, under the null hypothesis that data in the compared series are samples from distributions with equal medians, against the alternative that they are not. A significance level of $\alpha = 0.05$ was used. The *p*-values are reported in Table 4, where statistically significant differences are highlighted with *p*-values in bold. As one can clearly see from this table, the results discussed above on training data are statistically significant for the large majority of the considered benchmarks and GP systems.

While the results achieved from the training instances are important, it is fundamental to assess the ability of the proposed GP frameworks to handle unseen instances. The results achieved from the test instances are displayed in Fig. 5(b), (d), (f), and (h). For the bioavailability dataset (Fig. 5(b)), it is possible to observe that NAGP and NAGP_50 are clearly the best performers, able to produce comparable results. GSGP_LS outperforms GSGP, producing better quality models. Moving to the concrete dataset (Fig. 5(d)), NAGP_50 is the best performer, followed by GSGP_LS and NAGP. Also for this dataset, the poorest performance was obtained with GSGP. Hence, it is important to underline that the good performance we observed on the training instances for NAGP_50 does not translate into a lack of generalization ability (i.e., poor performance on the test set). Taking into account the energy dataset (Fig. 5(f)), NAGP_50 is the best performer, followed by NAGP, GSGP_LS, and GSGP. The same trend is observable in the PPB dataset (Fig. 5(h)). All in all, NAGP and NAGP_50 produce good-quality solutions that outperform GSGP in all the considered benchmarks. Additionally, as discussed in the first part of this section, results suggest that coupling the nested tournament selection of NAGP with GSGP is beneficial for achieving good-quality solutions that are also able to generalize over unseen instances.

As was done for the training instances, a statistical validation of the results was performed to assess the significance of the results achieved on the test set. The Lilliefors test showed that the data are not normally distributed and hence, the Wilcoxon rank-sum test for pairwise data comparison with Bonferroni correction was used. The null hypothesis of the test is that data in the compared series are samples from distributions with equal medians, against the alternative that they are not. A significance level of $\alpha = 0.05$ was used. The *p*-values are reported in Table 4, where statistically significant differences are highlighted with *p*-values in bold. As one can clearly see from this table, NAGP_50 obtained significantly better results with respect to the other competi-

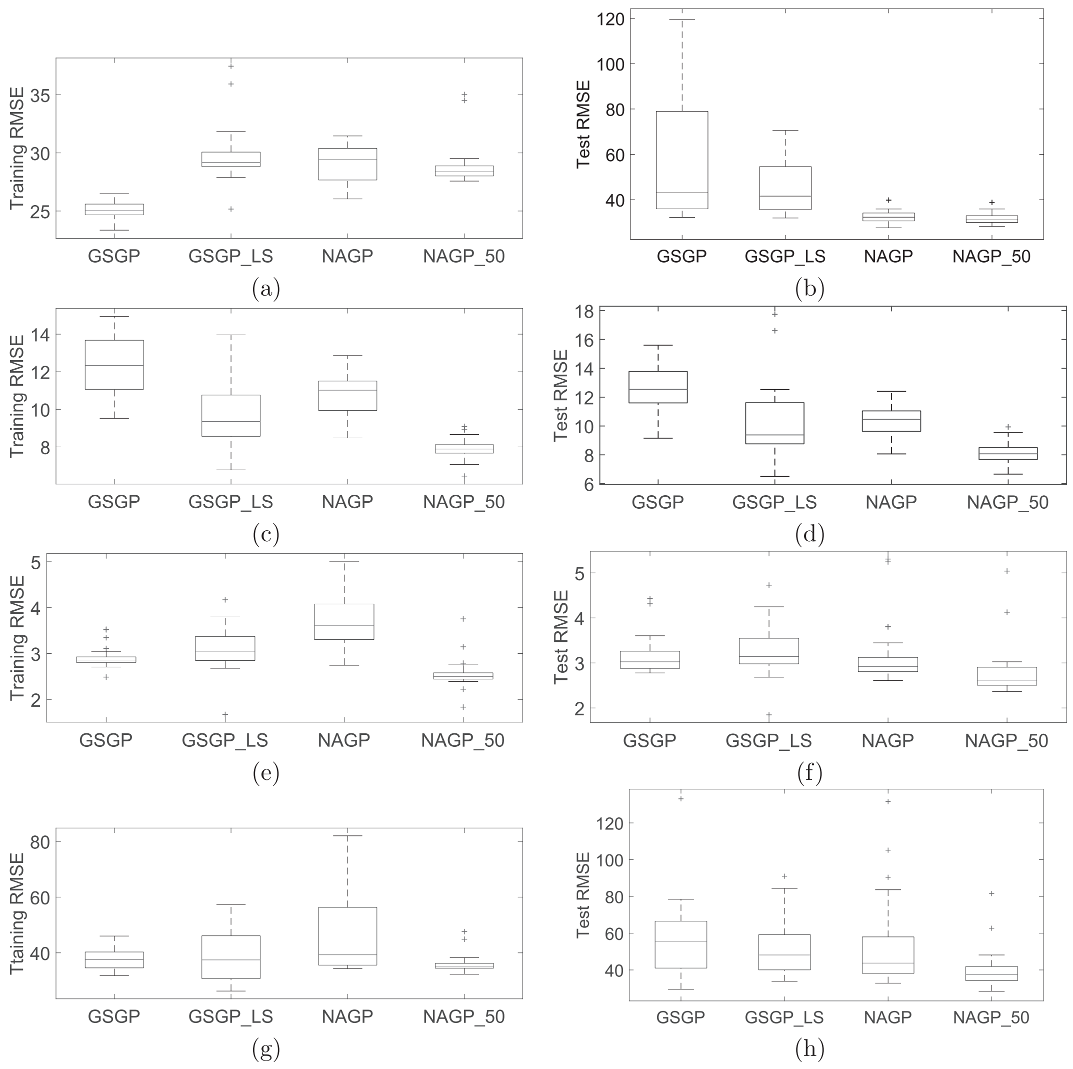


Fig. 5. Training and test errors for the considered benchmark problems. In each box, the central mark is the median, the edges of the box are the 25th and 75th percentiles, and the whiskers extend to the most extreme data points not considered outliers. The first line (i.e., plots (a), (b)) reports the results for the bioavailability dataset. The second line (i.e., plots (c), (d)) reports the results for the concrete dataset. The third line (i.e., plots (e), (f)) reports the results for the energy dataset. The fourth line (i.e., plots (g), (h)) reports the results for the PPB dataset.

tors on all the benchmarks, with the exception of the bioavailability dataset, where NAGP performed comparably.

To conclude this section, it is also important to discuss the dimensions of the evolved programs (i.e., the number of nodes of the solutions). For all the benchmarks taken into account, GSGP, GSGP_LS, and NAGP_50 generate much larger individuals compared to NAGP. This was expected, given that generating large individuals is a known drawback of geometric semantic operators (GSOs) [6]. The fact that in the first 50 generations NAGP_50 does not use GSOs only partially limits the problem, simply delaying the code growth, that is, after generation 50, of a comparable importance as for GSGP. On the other hand, NAGP is able to generate individuals that are much smaller: After an initial phase in which the size of the individuals grows for NAGP as well, we observed that NAGP basically has no further code growth. The same behavior was observed in all the datasets under examination, and to show the general trend, we reported the evolution of the model's size for the energy dataset in Fig. 7.

Finally, it is important to point out that the introduction of NAGP and NAGP_50 has a negative impact on the running time, considering the introduction of a nested tournament and the time needed to calculate all the required information. Anyway, taking into account that the criteria used are based on simple calculation/analysis of information associated with the current run of the GP system, the increased running-time is acceptable. More specifically, in the case of NAGP, and depending to the problem considered, the running time can increase by an amount between 8% and 15% of the GSGP's running time. In the case of NAGP_50, the increase in terms of running time is negligible, because the nested tournament is only used in the first generations of the evolutionary process.

6.1. Other machine learning methods

This section compares the results of the proposed GP systems (NAGP and NAGP_50) with the ones achieved by other well-known machine learning and statistical techniques that are commonly used to tackle regression problems. This comparison allows for understanding whether the proposed GP-based systems are competitive with respect to other, non-evolutionary, techniques. To perform this experimental campaign, we relied on the implementations provided by the Weka public domain software [23]. As was done for the GP-based systems, a preliminary study was performed to tune the parameters of the considered techniques. This process was simplified by the functionalities provided by Weka, which enables easily running a grid search over the space of

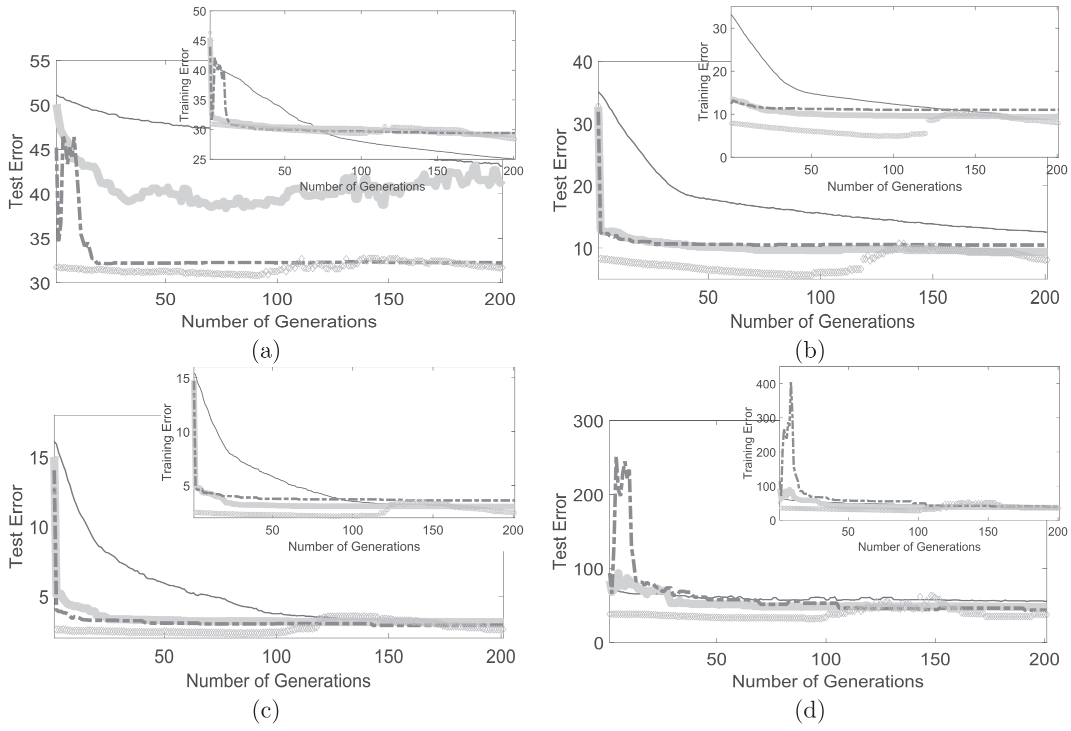


Fig. 6. Evolution of the (median) best fitness on the training (insets image) and test sets for the considered problems: (a) bioavailability; (b) concrete; (c) energy; (d) PPB. The legend for all the plots is — GSGP ▬ GSGP_LS - - NAGP_50 ◇ NAGP .

Table 4
The *p*-values returned by the Wilcoxon rank-sum test on training and test instances under the alternative hypothesis that the samples do not have equal medians. Bold denotes statistically significant values.

	BIOAVAILABILITY TRAINING				BIOAVAILABILITY TEST		
	GSGP_LS	NAGP	NAGP_50		GSGP_LS	NAGP	NAGP_50
GSGP	**1.02E-06**	**7.37E-07**	**3.23E-07**	GSGP	2.28E-01	**4.18E-05**	**1.36E-05**
GSGP_LS	–	6.24E-01	**1.42E-02**	GSGP_LS	–	**7.17E-05**	**1.81E-05**
NAGP		–	3.84E-01	NAGP		–	2.89E-01
	CONCRETE TRAINING				CONCRETE TEST		
	GSGP_LS	NAGP	NAGP_50		GSGP_LS	NAGP	NAGP_50
GSGP	**1.22E-06**	**1.48E-04**	**6.51E-11**	GSGP	**1.00E-05**	**1.22E-06**	**1.09E-10**
GSGP_LS	–	1.03E-02	**3.58E-06**	GSGP_LS	–	1.57E-01	**1.00E-05**
NAGP		–	**2.01E-10**	NAGP		–	**3.77E-09**
	ENERGY TRAINING				ENERGY TEST		
	GSGP_LS	NAGP	NAGP_50		GSGP_LS	NAGP	NAGP_50
GSGP	**2.23E-02**	**2.16E-08**	**8.81E-08**	GSGP	1.06E-01	2.71E-01	**5.46E-06**
GSGP_LS	–	**2.87E-05**	**9.60E-08**	GSGP_LS	–	**1.99E-02**	**1.61E-06**
NAGP		–	**7.35E-10**	NAGP		–	**5.26E-05**
	PPB TRAINING				PPB TEST		
	GSGP_LS	NAGP	NAGP_50		GSGP_LS	NAGP	NAGP_50
GSGP	1	8.23E-02	**1.91E-02**	GSGP	3.63E-01	2.71E-01	**2.01E-04**
GSGP_LS	–	3.51E-02	5.01E-01	GSGP_LS	–	5.40E-01	**8.14E-05**
NAGP		–	**1.49E-04**	NAGP		–	**2.20E-03**

the parameters.

The results of this comparison are reported in Fig. 8, where boxplots display the training and test fitness for the various methods and the benchmarks taken into account. In particular, in the figure, LIN stands for linear regression [24], MLP refers to feed-forward artificial neural networks [25] trained with the backpropagation learning rule [26], SVM refers to the support vector machines [27] with a polynomial kernel, and SQ stands for square regression [28].

According to these results, regarding the bioavailability problem (Fig. 8(a) and (b)), NAGP and NAGP_50 perform comparably with respect to MLP, while the best performer is LIN, followed by SVM. Regarding results in the unseen instances, NAGP_50 and NAGP were able to outperform all the other competitors. In particular, LIN and SVM showed some overfitting, and they were not able to achieve the same performance on the test set that was produced in the training instances. For the concrete dataset (Fig. 8(c) and (d)), MLP and NAGP_50 were the best performers in the training instances, followed by SVM and NAGP. On the other hand, MLP overfit the training instances and poorly generalized over unseen instances, as can be seen in Fig. 8(d). The best performer on the test set is NAGP_50, followed by NAGP. Considering the energy dataset (Fig. 8(e) and (f)), MLP was the best performer in the training instances, followed by NAGP_50, LIN, SVM, NAGP, and SQ. A similar behavior emerged from the fitness on the test set, where MLP was the best performer. NAGP_50 was the second best performer, fol-

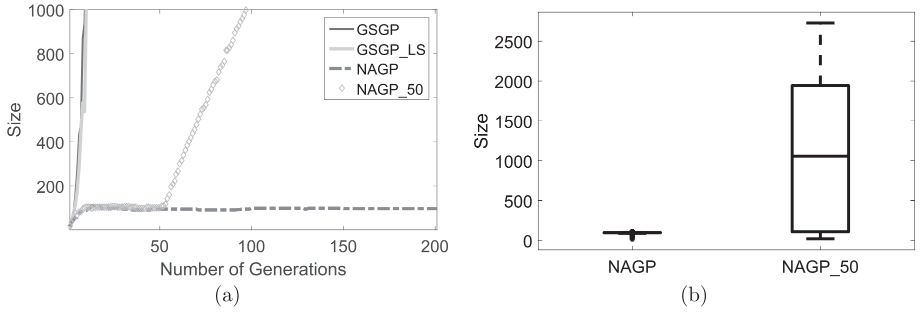


Fig. 7. Size of the best model for the four studied methods. Plot (a): evolution of the size of the individuals; plot (b): boxplots of the size of the best model at the end of the run. Results report the median of all the runs.

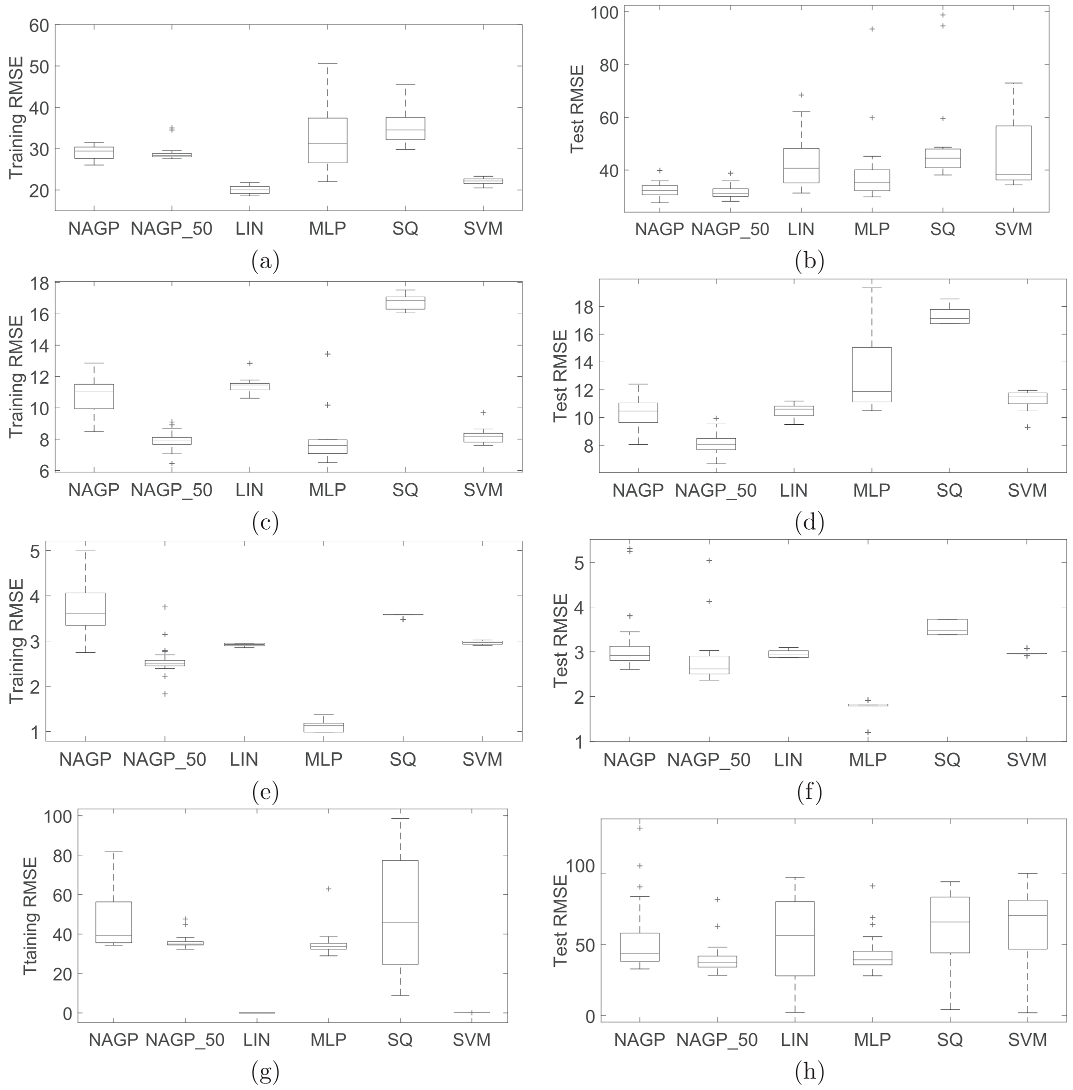


Fig. 8. Training and test error for the considered benchmark problems. In each box, the central mark is the median, the edges of the box are the 25th and 75th percentiles, and the whiskers extend to the most extreme data points not considered outliers. The first line (i.e., plots (a), (b)) reports the results for the bioavailability dataset. The second line (i.e., plots (c), (d)) reports the results for the concrete dataset. The third line (i.e., plots (e), (f)) reports the results for the energy dataset. The fourth line (i.e., plots (g), (h)) reports the results for the PPB dataset.

lowed by LIN and NAGP (which produced comparable results). Finally, on the PPB dataset (Fig. 8(g) and (h)), LIN and SVM were able to obtain the best performance in the training instances, with solutions that had a fitness close to zero. The remaining techniques produced larger RMSE values, but, as one can see from Fig. 8(h), the situation radically changed on the test set. In particular, LIN and SVM were not able to produce the same performance achieved on the training set and, over unseen instances, the best performer is NAGP_50, followed by MLP and NAGP. All in all, NAGP and NAGP_50 are able to produce competitive results with respect to other non-evolutionary techniques. More specifically, NAGP_50 was the best performer in three out of four benchmarks with respect to the fitness over unseen instances, and it was the second

Table 5
The *p*-values returned by the Wilcoxon rank-sum test on training instances under the alternative hypothesis that the samples do not have equal medians. Bold denotes statistically significant values.

	NAGP_50	LIN	MLP	SQR	SVM
BIOAVAILABILITY TRAINING					
NAGP	3.84E-01	**3.23E-07**	1.79E-01	**5.62E-06**	**3.23E-07**
NAGP_50	–	**3.23E-07**	4.20E-01	**6.53E-06**	**3.23E-07**
LIN	–	–	**3.23E-07**	**3.23E-07**	**5.62E-06**
MLP	–	–	–	1.17E-01	**5.62E-06**
SQR	–	–	–	–	**3.23E-07**
CONCRETE TRAINING					
NAGP	**2.01E-10**	1.52E-02	**2.40E-06**	**6.26E-11**	**1.94E-10**
NAGP_50	–	**6.26E-11**	1.16E-01	**6.26E-11**	3.56E-02
LIN	–	–	**2.13E-07**	**6.03E-11**	**6.03E-11**
MLP	–	–	–	**6.03E-11**	**2.80E-03**
SQR	–	–	–	–	**6.03E-11**
ENERGY TRAINING					
NAGP	**3.47E-10**	**4.62E-09**	**2.43E-11**	**4.18E-01**	**7.21E-09**
NAGP_50	–	**7.21E-09**	**2.43E-11**	**4.62E-10**	**7.21E-09**
LIN	–	–	**1.96E-11**	**1.96E-11**	**8.05E-04**
MLP	–	–	–	**1.96E-11**	**1.96E-11**
SQR	–	–	–	–	**1.96E-11**
PPB TRAINING					
NAGP	**1.49E-04**	**1.21E-12**	**8.84E-07**	6.63E-01	**2.97E-11**
NAGP_50	–	**1.21E-12**	1.50E-02	3.92E-02	**2.97E-11**
LIN	–	–	**1.21E-12**	**1.21E-12**	**1.19E-12**
MLP	–	–	–	**1.27E-02**	**2.97E-11**
SQR	–	–	–	–	**2.97E-11**

Table 6
The *p*-values returned by the Wilcoxon rank-sum test for unseen instances under the alternative hypothesis that the samples do not have equal medians. Bold denotes statistically significant values.

	NAGP_50	LIN	MLP	SQR	SVM
BIOAVAILABILITY TEST					
NAGP	2.89E-01	**5.97E-04**	3.82E-02	**7.37E-07**	**2.40E-05**
NAGP_50		**7.17E-05**	**4.20E-03**	**4.50E-07**	**8.78E-06**
LIN			9.67E-02	2.35E-01	5.80E-01
MLP				**5.97E-04**	2.37E-02
SQR					4.80E-02
CONCRETE TEST					
NAGP	**3.77E-09**	8.89E-01	**1.63E-06**	**6.26E-11**	**2.39E-04**
NAGP_50	–	**1.95E-10**	**6.26E-11**	**6.26E-11**	**1.58E-10**
LIN	–	–	**2.98E-07**	**6.06E-11**	**4.79E-05**
MLP	–	–	–	**9.83E-05**	3.55E-02
SQR	–	–	–	–	**6.03E-11**
ENERGY TEST					
NAGP	**5.26E-05**	9.70E-01	**2.43E-11**	**4.27E-06**	5.88E-01
NAGP_50	–	**4.55E-05**	**2.43E-11**	**7.21E-09**	**1.79E-06**
LIN	–	–	**1.96E-11**	**1.96E-11**	4.60E-01
MLP	–	–	–	**1.96E-11**	**1.96E-11**
SQR	–	–	–	–	**1.96E-11**
PPB TEST					
NAGP	**2.20E-03**	1	3.78E-02	9.93E-02	3.78E-02
NAGP_50	–	1.54E-01	2.84E-01	**1.30E-03**	**3.18E-04**
LIN	–	–	2.71E-01	4.83E-01	1.96E-01
MLP	–	–	–	**5.10E-03**	**7.30E-04**
SQR	–	–	–	–	5.90E-01

best performer when the energy dataset was considered.

The same statistical validation previously performed for the GP-based methods was adopted to assess the results achieved by the other non-evolutionary techniques taken into account, and the *p*-values are reported in Table 5 (for the training sets) and in Table 6 (for the test sets). Bold values suggest that the differences in terms of performance between the proposed NAGP_50 system and the other techniques are statistically significant for the majority of the benchmarks.

6.2. Discussion of an evolved model

In this section, we show and discuss the best multi-individual evolved by NAGP in our simulations for the concrete dataset. It is important to point out that, as the results reported above clearly show,

Table 7
Number of occurrences of each variable in the expressions presented in Section 6.2.

Variable	*X*0	*X*1	*X*2	*X*3	*X*4	*X*5	*X*6	*X*7
Number of Occurrences	6	9	7	7	7	4	14	13

this would not be possible for NAGP_50 or GSGP, because these two methods use geometric semantic operators which cause rapid growth in the size of the evolved solutions. The best multi-individual evolved by NAGP in all the runs that we have performed for the concrete problem was composed by the following expressions, in prefix notation:

- Expression 1: (∗ (+(∗X0 (- X6(∗ (+(- (/(∗X3 (- −0.75 X5)) (/(/0.5(+−1.0 -0.25)) (/−1.0 0.75))) (- X1 (- (/X7 0.25)(+(∗ (/−0.75 (+X7 X0)) (∗ (- 1.0 X6) (- (∗ (/(+X2 X1) −0.25) (+(/X7 (+−0.5 (/(/X4 (+−1.0 X1)) X7)))(- (- 0.25 1.0) (+(+X7 X1) (- 1.0-0.25))))) 0.5)))0.5))))(- (/(- (+−0.5 (∗ −0.25 0.75))(/(∗(/X5(- −0.25 0.5)) (+X0 (- (/(- −0.25 X3) (/(/(/(/(- (∗(/X2 -1.0)X0) X6) −0.25) (/(/(+−0.25 -1.0) (+X7 -0.5)) (+−0.25 -0.75)))X1) X1))−0.75)))(/(/(/(+X6 (/0.25(∗ −0.75 1.0))) 0.75)(+−1.0 (+1.0 X3))) −1.0))) (/X6 (/(+ (+X6 -1.0)(∗ −0.5 (- 1.0 (- −0.75 X3)))) 0.75))) −0.25))(/X7 (∗ (+−0.75 (/1.0 (∗(/X6 (/0.75(+0.75 X4))) (∗ (- (+−0.25 X4) 0.75) (∗X6 (/(∗ (/−0.25 (+(- −0.5 X6) 1.0)) (- X6 0.25)) X7)))))) X7))))) (∗X7 -0.5)) 34.0) X5) 33.0) 36.0) X6) 23.0) 31.0) 23.0) 20.0) 31.0) 39.0) 39.0) (- X6 1.0)) 36.0) 28.0) 34.0) 22.0) 25.0) 38.0) 26.0) 29.0) 34.0) 27.0) 30.0) 23.0) 33.0) 35.0) 24.0) 34.0) 36.0) 36.0) 37.0) 38.0) 36.0) 27.0) 39.0) 36.0) 20.0) 34.0) 37.0) 37.0) 37.0) 36.0) 32.0) 37.0) 39.0) 33.0) 26.0) 39.0) 31.0) 33.0) 24.0) 27.0) 27.0) 33.0) 39.0) 37.0) 38.0) 36.0) 32.0) 23.0) 35.0) 24.0) 39.0) 26.0) 26.0) (+(+X0 (∗ (- 1.0 (+(/X2 0.75) −0.75)) −0.5)) (+(∗X1 1.0) X7))) 26.0) 37.0) 37.0) 27.0) 32.0) 38.0) 22.0) 37.0) 34.0) 31.0) 28.0) 30.0) 21.0) 26.0) 23.0) 20.0) 38.0) 38.0) 33.0) 32.0) 21.0) 24.0) 20.0) 37.0) 30.0) 21.0)
- Expression 2: (∗ (∗ (/X4 (+X7 (+(∗ (- (- (/(∗ (- (/ (+1.0 X3) (- X1 (- (- X2 X6) −0.75))) (- (/(- X2 X4) (∗ (+(/(+−1.0 0.25) (+0.25 X0)) (+X3 (- X3 0.5))) X4)) X4)) X1) (/0.75-0.25)) (- X5 -1.0)) (+−0.5 0.5)) (/−1.0 X2)) (- X6 X2)))) X7) 21.0)

Table 2 includes a reference to the different variables used in this expression (only the IDs - *X*0, *X*1, …, *X*7 – referenced in the table are used in the above expressions). If we consider the reconstructed expression P_{opt} (as in Equation (2)) using these two expressions, P_{opt} has an error on the training set equal to 9.53 and an error on the test set equal to 9.06. Both the relationship between the training and test error (they have the same order of magnitude and the error on the test set is even smaller) and a comparison with the median results reported previously allow us to conclude that this solution has a very good performance, with no overfitting. The first thought that comes to mind when viewing these two expressions is that the first one is significantly different from the second one: first of all in terms of size (the first expression is clearly larger than the second), but also in terms of tree shape. Observing the first expression, in fact, one may notice a sort of skewed and unbalanced shape consisting of several multiplications by constant numbers. This observation is not surprising: the first of these two expressions, in fact, is the one that has undergone multiplication by the constant λ during the mutation events, as explained previously. These continuous multiplications by constants also have, of course, an impact on the size of the expression (this is the reason why the first expression is larger than the second). However, it is easy to understand that all these multiplications by a constant can be easily simplified (i.e., transformed into one single multiplication by a constant). Concerning the second expression, instead, we can see that it is much simpler and quite easy to read (numeric simplifications are also possible in this second expression, which would make it even simpler and easier to read). Concerning the variables used by the two models, Table 7 shows the number of times that each of the variables appears in these two expressions. From this table, we can see that variables *X*6 and *X*7 are the ones that appear most frequently in the expressions, and thus we hypothesize that these variables are the most useful (i.e., informative) by NAGP for the correct reconstruction of the target. These variables represent fine aggregate (expressed in kg/m3) and age of testing (expressed in number of days), respectively.

Interpretability is a delicate subject in Machine Learning. In many possible different applications, accuracy can be considered as the most important characteristics that a model must have. But in several cases, models are used as basis for decision making, and as such must be trusted. Interpretability of the model is at the basis of trust, in many cases. However, interpretability is subjective and extremely hard to formally define. In the case of the model reported above, as usual in GP, interpretability is only partial: the expressions, although much smaller than the ones generated by NAGP_50 and GSGP, are still too large and complex to be completely understood in all their details. Nevertheless, it is possible to understand several characteristics of the model: the features it contains, the relative importance of the employed features and the primitive operators used to compose those features. These characteristics are expected to give important insights to application experts.

7. Conclusions and future work

Previous work has shown the success of two sophisticated GP systems, able to exploit semantic awareness, for generating predictive models in many applicative fields. These two systems, GSGP and GSGP_LS, have clearly outperformed many of the existing methods, thus fostering themselves as the state-of-the-art technology for the generation of predictive models in those applicative areas. Nevertheless, GSGP and GSGP_LS have the important drawback of generating predictive models that, although very accurate, have an extremely large size, which makes them impossible to interpret by humans. In other words, GSGP and GSGP_LS are "black box" methods. In this paper, we introduced a novel GP system, called NAGP, and a variant of it, called NAGP_50. Both NAGP and NAGP_50 significantly outperform GSGP and GSGP_LS in terms of prediction accuracy. Furthermore, NAGP (even though it is outperformed by NAGP_50 in terms of prediction accuracy) is able to generate predictive models that are much smaller, compared to the other studied methods. The predictive models generated by NAGP, composed by approximately only 50 tree nodes, are small enough to be read and understood by a human. Compared to GSGP and GSGP_LS, NAGP exploits semantic awareness in a completely different way. More specifically, NAGP is based on the concept of alignment in the error space, which bestows on NAGP a larger freedom to explore the search space, looking for appropriate solutions. This greater versatility of NAGP, compared to GSGP and GSGP_LS, is probably the reason NAGP was able to outperform the other two methods. The studied applications, in fact, are very complex, and better exploration power is beneficial.

Nevertheless, a lot of future work is still in demand to further improve the method and its accuracy. In particular, one of the most important limitations of this paper is that only alignments in two dimensions are considered. In other words, NAGP evolves individuals that are pairs of programs and so NAGP is only able to search for pairs of optimally aligned programs. Our current research is focused on extending the method to more than two dimensions. For instance, we are currently working on the development of systems that evolve individuals that are triplets of programs, aimed at finding triplets of optimally coplanar individuals. The subsequent step will be to further extend the method, possibly generalizing to any number of dimensions. The design of self-configuring methods, which automatically decide the most appropriate dimension, is one of the most ambitious goals of our current work. Last but not least, the study of several possible types of crossover for NAGP is definitely part of our planned future work.

Acknowledgments

This work was funded by CONACYT (Mexico) project no. FC-2015-2/944 “Aprendizaje evolutivo a gran escala” and TecNM (Mexico) project no. 6826-18-p.

Appendix A. Supplementary data

Supplementary data to this article can be found online at https://doi.org/10.1016/j.swevo.2018.09.006.

References

[1] T. Bäck, D.B. Fogel, Z. Michalewicz, Handbook of Evolutionary Computation, CRC Press, 1997.

[2] C. Darwin, On the Origin of Species, 1859, Routledge, 2004.

[3] J.R. Koza, Genetic Programming: on the Programming of Computers by Means of Natural Selection, MIT Press, Cambridge, MA, USA, 1992.

[4] J.R. Koza, Human-competitive results produced by genetic programming, Genet. Program. Evolvable Mach. 11 (2010) 251–284.

[5] L. Vanneschi, M. Castelli, S. Silva, A survey of semantic methods in genetic programming, Genet. Program. Evolvable Mach. 15 (2014) 195–214.

[6] A. Moraglio, K. Krawiec, C. Johnson, Geometric semantic genetic programming, in: C. Coello, V. Cutello, K. Deb, S. Forrest, G. Nicosia, M. Pavone (Eds.), Parallel Problem Solving from Nature - PPSN XII, Volume 7491 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2012, pp. 21–31.

[7] M. Castelli, L. Trujillo, L. Vanneschi, S. Silva, E. Z-Flores, P. Legrand, Geometric semantic genetic programming with local search, in: Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation, GECCO ’15, ACM, New York, NY, USA, 2015, pp. 999–1006.

[8] M. Castelli, L. Vanneschi, M.D. Felice, Forecasting short-term electricity consumption using a semantics-based genetic programming framework: the south Italy case, Energy Econ. 47 (2015) 37–41.

[9] M. Castelli, L. Vanneschi, L. Manzoni, A. Popovič, Semantic genetic programming for fast and accurate data knowledge discovery, Swarm Evol. Comput. 26 (2016) 1–7.

[10] S. Ruberto, L. Vanneschi, M. Castelli, S. Silva, Esagp – a semantic GP framework based on alignment in the error space, in: M. Nicolau, K. Krawiec, M.I. Heywood, M. Castelli, P. García-Sánchez, J.J. Merelo, V.M. Rivas Santos, K. Sim (Eds.), European Conference on Genetic Programming, Springer, 2014, pp. 150–161.

[11] M. Castelli, L. Vanneschi, S. Silva, S. Ruberto, How to exploit alignment in the error space: two different GP models, in: R. Riolo, W.P. Worzel, M. Kotanchek (Eds.), Genetic Programming Theory and Practice XII, Genetic and Evolutionary Computation, Springer, Ann Arbor, USA, 2014, pp. 133–148.

[12] I. Gonçalves, S. Silva, C.M. Fonseca, M. Castelli, Arbitrarily close alignments in the error space: a geometric semantic genetic programming approach, in: Proceedings of the 2016 on Genetic and Evolutionary Computation Conference Companion, GECCO ’16 Companion, ACM, New York, NY, USA, 2016, pp. 99–100.

[13] J. Nalepa, Z.J. Czech, New selection schemes in a memetic algorithm for the vehicle routing problem with time windows, in: International Conference on Adaptive and Natural Computing Algorithms, Springer, 2013, pp. 396–405.

[14] K. Diwold, A. Aderhold, A. Scheidler, M. Middendorf, Performance evaluation of artificial bee colony optimization and new selection schemes, Mem. Comput. 3 (2011) 149–162.

[15] A. Tsanas, A. Xifara, Accurate quantitative estimation of energy performance of residential buildings using statistical machine learning tools, Energy Build. 49 (2012) 560–567.

[16] M. Castelli, L. Trujillo, L. Vanneschi, A. Popovič, Prediction of energy performance of residential buildings: a genetic programming approach, Energy Build. 102 (2015) 67–74.

[17] I.-C. Yeh, Modeling of strength of high-performance concrete using artificial neural networks, Cement Concr. Res. 28 (1998) 1797–1808.

[18] M. Castelli, L. Vanneschi, S. Silva, Prediction of high performance concrete strength using genetic programming with geometric semantic genetic operators, Expert Syst. Appl. 40 (2013) 6856–6862.

[19] L. Vanneschi, M. Castelli, I. Gonçalves, L. Manzoni, S. Silva, Geometric semantic genetic programming for biomedical applications: a state of the art upgrade, in: Evolutionary Computation (CEC), 2017 IEEE Congress on, IEEE, pp. 177–184.

[20] L. Vanneschi, S. Silva, M. Castelli, L. Manzoni, Geometric semantic genetic programming for real life applications, in: Genetic Programming Theory and Practice XI, Springer, 2014, pp. 191–209.

[21] L. Vanneschi, M. Castelli, L. Manzoni, S. Silva, A new implementation of geometric semantic GP and its application to problems in pharmacokinetics, in: Proceedings of the 16th European Conference on Genetic Programming, EuroGP 2013, Volume 7831 of LNCS, Springer Verlag, Vienna, Austria, 2013, pp. 205–216.

[22] J. Zhang, W.-N. Chen, Z.-H. Zhan, W.-J. Yu, Y.-L. Li, N. Chen, Q. Zhou, A survey on algorithm adaptation in evolutionary computation, Front. Electr. Electron. Eng. 7 (2012) 16–31.

[23] Weka Machine Learning Project, Weka, 2018, http://www.cs.waikato.ac.nz/%7eml/weka.

[24] S. Weisberg, Applied Linear Regression, Wiley Series in Probability and Statistics, Wiley, 2005.

[25] S. Haykin, Neural Networks: a Comprehensive Foundation, Prentice Hall, 1999.

[26] K. Gurney, An Introduction to Neural Networks, Taylor & Francis, 1997.

[27] B. Schölkopf, A. Smola, Learning with Kernels: Support Vector Machines, Regularization, Optimization, and beyond, Adaptive Computation and Machine Learning, MIT Press, 2002.

[28] R.E. Kass, Nonlinear regression analysis and its applications, J. Am. Stat. Assoc. 85 (1990) 594–596.