Expert Systems with Applications xxx (2013) xxx–xxx

Contents lists available at SciVerse ScienceDirect

Expert Systems with Applications

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

# Prediction of high performance concrete strength using Genetic Programming with geometric semantic genetic operators

Mauro Castelli [a,*], Leonardo Vanneschi [a], Sara Silva [b]

[a] ISEGI, Universidade Nova de Lisboa, 1070-312 Lisboa, Portugal
[b] INESC-ID, IST/ Universidade Técnica de Lisboa, 1000-029 Lisboa, Portugal

ARTICLE INFO

*Keywords:*
High performance concrete
Strength prediction
Artificial intelligence
Genetic Programming
Geometric operators
Semantics

ABSTRACT

Concrete is a composite construction material made primarily with aggregate, cement, and water. In addition to the basic ingredients used in conventional concrete, high-performance concrete incorporates supplementary cementitious materials, such as fly ash and blast furnace slag, and chemical admixture, such as superplasticizer. Hence, high-performance concrete is a highly complex material and modeling its behavior represents a difficult task. In this paper, we propose an intelligent system based on Genetic Programming for the prediction of high-performance concrete strength. The system we propose is called Geometric Semantic Genetic Programming, and it is based on recently defined geometric semantic genetic operators for Genetic Programming. Experimental results show the suitability of the proposed system for the prediction of concrete strength. In particular, the new method provides significantly better results than the ones produced by standard Genetic Programming and other machine learning methods, both on training and on out-of-sample data.



## 1. Introduction

Concrete is a composite construction material made primarily with aggregate, cement, and water. There are many formulations of concrete, which provide varied properties, and concrete is the most-used man-made product in the world (Lomborg, 2001). Modern concrete mix designs can be complex. The choice of a concrete mix depends on the need of the project both in terms of strength and appearance and in relation to local legislation and building codes. The design begins by determining the requirements of the concrete. These requirements take into consideration the weather conditions that the concrete will be exposed to in service, and the required design strength. Many factors need to be taken into account, from the cost of the various additives and aggregates, to the trade offs between, the "slump" for easy mixing and placement and ultimate performance. A mix is then designed using cement, coarse and fine aggregates, water and chemical admixtures. The method of mixing will also be specified, as well as conditions that it may be used in. This allows a user of the concrete to be confident that the structure will perform properly. As reported in Yeh (1998), high-performance concrete (HPC) is a new terminology used in the concrete construction industry. In addition to the basic ingredients in conventional concrete the making of HPC needs to incorporate supplementary cementitious materials, such as fly ash and blast furnace slag, and chemical admixture, such as superplasticizer (Kumar, Singh, & Singh, 2012). High-performance concrete is such a highly complex material that modeling its behavior is a difficult task.

The Abrams' water-to-cement ratio (w/c) law (Abrams, 1927; Nagaraj & Banu, 1996) has been described as the most useful and significant advancement in the history of concrete technology. According to Abrams's law, the compressive strength of concrete varies inversely with the W/C ratio. Hence, an increase in the w/c decreases the concrete strength, whereas a decrease in the w/c ratio increases the strength. The implication, therefore, is that the strengths of various but comparable concrete are identical as long as their w/c ratios remain the same, regardless of the details of the compositions.

The Abrams rule implies that only the quality of the cement paste controls the strength of comparable cement. The paste quantity does not matter. Analysis of a variety of experimental data shows that this is not quite true (Popovics, 1990). For instance, if two comparable concrete mixtures have the same w/c ratio, the strength of the concrete with the higher cement content is lower (Popovics, 1990).

As reported in Yeh (1998), several studies independently have shown that concrete strength development is determined not only by the w/c ratio, but that it is also influenced by the content of other ingredients (Bhanja & Sengupta, 2005). Therefore, although experimental data have shown the practical acceptability of this rule within wide limits, a few deviations have been reported. The current empirical equations for estimating compressive strength

* Corresponding author. Tel.: +351 393384137740.
*E-mail addresses:* mcastelli@isegi.unl.pt (M. Castelli), lvanneschi@isegi.unl.pt (L. Vanneschi), sara@kdbio.inesc-id.pt (S. Silva).

0957-4174/$ - see front matter © 2013 Published by Elsevier Ltd.
http://dx.doi.org/10.1016/j.eswa.2013.06.037

are based on tests of concrete without supplementary cementitious materials. The validity of these relationships for concrete with supplementary cementitious materials (fly ash, blast furnace slag, etc.) should be investigated (Bhanja & Sengupta, 2005). The more we know about the concrete composition versus strength relationship, the better we can understand the nature of concrete and how to optimize the concrete mixture.

All these aspects highlight the need of reliable and accurate techniques that allow modeling the behavior of concrete materials.

In this paper, for the first time, we propose an intelligent system based on Genetic Programming for the prediction of the concrete strength. More in detail, in this work we use recently defined geometric semantic genetic operators for Genetic Programming. These operators allow to include the concept of semantics in the search process and have several advantages with respect to standard genetic operators used in Genetic Programming.

The paper is organized as follows: Section 2 introduces basic concepts about Genetic Programming with a particular focus on the standard operators used in the evolutionary process; Section 3 presents the geometric semantic operators used in this work and outlines some of their properties that have a direct effect on the search process; Section 4 presents a literature review on using computational intelligence methods in simulating the behavior of concrete materials; Section 5 describes the data used in this work, the experimental settings and proposes an accurate discussion and analysis of the results; Section 6 concludes the paper summarizing the results that have been achieved.

## 2. Genetic Programming

Models lie in the core of any technology in any industry, be it finance, manufacturing, services, mining, or information technology. The task of data-driven modeling lies in using a limited number of observations of system variables for inferring relationships among these variables. The design of reliable learning machines for data-driven modeling tasks is of strategic importance, as there are many systems that cannot be accurately modeled by classical mathematical or statistical techniques. Reliable learning in the field of Machine Learning (ML) revolves around the notion of generalization, which is the ability of a learned model to correctly explain data that are drawn from the same distribution as training data, but have not been presented during the training process, and it is this very important property that ML algorithms aim to optimize. Genetic Programming (GP) (Koza, 1992; Poli, Langdon, & Mcphee, 2008) is one of the youngest paradigms inside the computational intelligence research area called Evolutionary Computation (EC) and consists in the automated learning of computer programs by means of a process mimicking Darwinian evolution. GP tackles learning problems by means of searching a computer program space for the program that better respects some given functional specifications.GP is an evolutionary computation technique that automatically solves problems without requiring the user to know or specify the form or structure of the solution in advance. At the most abstract level GP is a systematic, domain-independent method for getting computers to solve problems automatically starting from a high-level statement of what needs to be done. In GP a population of computer programs is evolved. That is, generation by generation, GP stochastically transforms populations of programs into new, hopefully better, populations of programs. The search is performed using an EA. The recipe for solving a problem with GP is as follows:

- Chose a representation space in which candidate solutions can be specified. This consists of deciding on the primitives of the programming language that will be used to construct programs. A program is built up from a terminal set (the variables in the problem) and a function set (the basic operators).
- Design the fitness criteria for evaluating the quality of a solution. This involves the execution of a candidate solution on a suite of test cases, reminiscent of the process of black-box testing. In case of supervised learning, a distance-based function is employed to quantify the divergence of a candidate's behavior from the desired one.
- Design a parent selection and replacement policy. Central to every EA is the concept of fitness-driven selection in order to exert an evolutionary pressure towards promising areas of the program space. The replacement policy determines the way in which newly created offspring programs replace their parents in the population.
- Design a variation mechanism for generating offspring from a parent or a set of parents. Standard GP uses two main variation operators: crossover and mutation. Crossover recombines parts of the structure of two individuals, whereas mutation stochastically alters a portion of the structure of an individual.
- After a random initialization of a population of computer programs, an iterative application of selection-variation-replacement is employed to improve the programs quality in a stepwise refinement way.

In GP learning, both terms of program and model refer to the same entity and will be used interchangeably. Supervised learning of regression models in GP relies on the extraction of implicit relationships that may exist in the input–output mappings specified by the training examples. The discovered relationships are expressed in symbolic form, which is traditionally represented by an expression-tree structure. For a complete introduction to GP the reader is referred to Koza (1992). In this work we used genetic operators that, diversely from the canonical ones, are based on the concept of semantics that will be introduced in the next section. To understand the differences between the genetic operators used in this work (described in Section 3) and the ones used in the standard GP algorithm, the latter are briefly described.

*The "Standard" Crossover Operator*. The crossover operator is traditionally used to combine the genetic material of two parents by swapping a part of one parent with a part of the other. More specifically, tree-based crossover proceeds by the following steps:

- Choose two individuals as parents, based on mating selection policy (the better the fitness of the individual the higher its probability of being selected).
- Select a random subtree in each parent. The selection of subtrees can be biased so that subtrees constituting terminals are selected with lower probability than other subtrees.
- Swap the selected subtrees between the two parents. The resulting individuals are the children.

This process is shown, using two arbitrary simple Boolean expressions as parents, in Fig. 1.

*The Mutation Operator*. The mutation operation introduces random changes in the structures of the individuals in the population. While there are many different mutation operators, here subtree mutation (that is the most well-known one and also the one used in this work) is described. Subtree mutation begins by selecting a point at random within the "parent" tree. Then, it removes whatever is currently at the selected point and whatever is below the selected point and inserts a randomly generated subtree at that point. This operation is controlled by a parameter that specifies the maximum size (usually measured in terms of tree depth) for the newly created subtree that is to be inserted. This parameter typically has the same value as the parameter for the maximum size of trees in the initial population. This process is shown in Fig. 2, where an arbitrary simple Boolean expression is mutated by replacing one of its subtrees with a random tree.



Q1 Please cite this article in press as: Castelli, M., et al. Prediction of high performance concrete strength using Genetic Programming with geometric semantic genetic operators. *Expert Systems with Applications* (2013), http://dx.doi.org/10.1016/j.eswa.2013.06.037

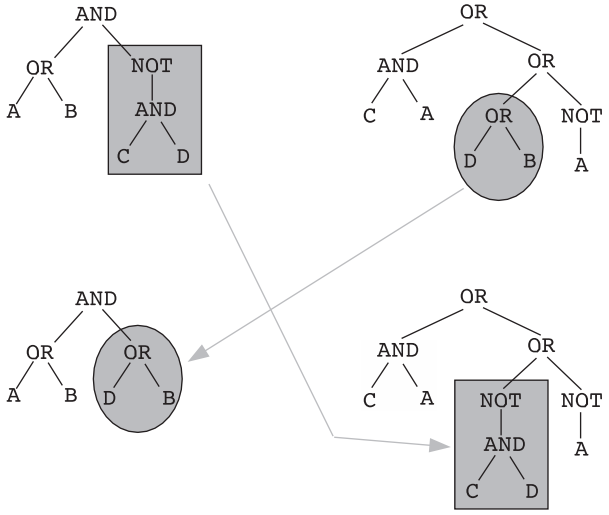


**Fig. 1.** An example of standard GP (subtree swapping) crossover.

## 3. Geometric semantic operators

In the last few years, GP has been extensively used to address problems in different domains (Aslam, Zhu, & Nandi, 2013; Gusel & Brezocnik, 2011; Guo, Rivero, Dorado, Munteanu, & Pazos, 2011; Kayadelen, 2011; Tsakonas, Dounias, Doumpos, & Zopounidis, 2006) and it has produced a wide set of results that have been defined human-competitive (Koza, 2010). While these results have demonstrated the suitability of GP in tackling real-life problems, research has recently focused on developing new variants of GP in order to further improve its performances. In particular, efforts have been dedicated to an aspect that was only marginally considered up to some years ago: the definition of methods based on the semantics of the solutions (Beadle & Johnson, 2008; Beadle & Johnson, 2009; Castelli et al.; Krawiec & Lichocki, 2009; Jackson, 2010). Although there is no universally accepted definition of semantics in GP, this term often refers to the behavior of a program, once it is executed on a set of data. For this reason, in many references, including here, the term semantics is intended as the vector of outputs a program produces on the training data (Moraglio et al., 2012). Thought semantics determines what a program actually does, the traditional GP operators, like crossover and mutation described so far ignore this knowledge and manipulate programs only considering their syntax. Abstraction from semantics allows them to rely on simple, generic search operators. The main consequence of this choice is that it is difficult to predict how modifications of programs will affect their semantics. Very recently, new genetic operators, called geometric semantic genetic operators have been proposed in Moraglio et al. (2012). These operators, that manipulate programs considering directly their semantic information, have a number of theoretical advantages, compared to the ones of standard GP, the most important one being the fact that they induce a unimodal fitness landscape (Stadler & Institute, 1995) on any problem consisting in finding the match between a set of input data and a set of expected target ones. According to the theory of fitness landscapes (Vanneschi, 2004) this should relevantly improve GP evolvability (Altenberg, 1994) (i.e. the ability of genetic operators to produce offspring that are fitter than their parents) on all these problems. In this section we report the definition of geometric semantic operators for real functions domains, since these are the operators we will use in the experimental phase.

**Definition 1** (*Geometric Semantic Crossover*). Given two parent functions $T_1, T_2 : \mathbb{R}^n \to \mathbb{R}$, the geometric semantic crossover returns the real function $T_{XO} = (T_1 \cdot T_R) + ((1 - T_R) \cdot T_2)$, where $T_R$ is a random real function whose output values range in the interval [0,1].

The interested reader is referred to Moraglio et al. (2012) for a formal proof of the fact that this operator corresponds to a geometric crossover on the semantic space, in the sense that it produces an offspring that stands between its parents in this space. Nevertheless, even without a formal proof, we can have an intuition of it by considering that the (unique) offspring generated by this crossover has a semantic vector that is a linear combination of the semantics of the parents with random coefficients included in [0,1] and whose sum is equal to 1. The authors of Moraglio et al. (2012) also prove an interesting consequence of this fact: the fitness of the offspring cannot be worse than the fitness of the worst of its parents. Also in this case we do not replicate the proof here, but we limit ourselves to give a visual intuition of this property: in Fig. 3 we report a simple two-dimensional semantic space in which we draw a target function $T$ (training points are represented by "X" symbols), two parents $P_1$ and $P_2$ and one of their offspring $O$ (which by construction is included between its par-

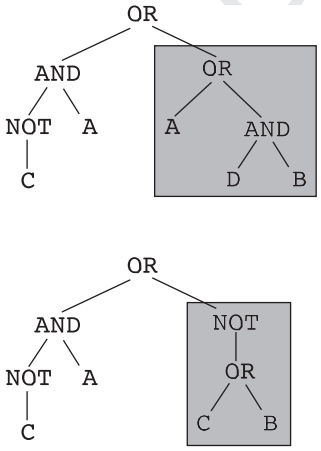


**Fig. 2.** An example of standard GP (subtree) mutation.

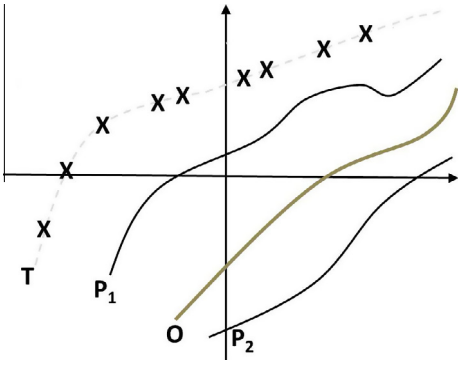


**Fig. 3.** The toy functions that we use to give a visual intuition of the fact that geometric semantic crossover produces an offspring that is at least not worse than the worst of its parents. In this simple case, offspring $O$ (which stands between parents $P_1$ and $P_2$ by construction) is clearly closer to target $T$ (training points represented by "X" symbols) than parent $P_2$.

Q1 Please cite this article in press as: Castelli, M., et al. Prediction of high performance concrete strength using Genetic Programming with geometric semantic genetic operators. *Expert Systems with Applications* (2013), http://dx.doi.org/10.1016/j.eswa.2013.06.037

ents). In Fig. 3, it is immediately clear that $O$ is closer to $T$ than $P_2$ (which is the worst parent in this case). The generality of this property is proven in Moraglio et al. (2012).

To constrain $T_R$ in producing values in [0,1] we use the sigmoid function: $T_R = \frac{1}{1+e^{-T_{rand}}}$ where $T_{rand}$ is a random tree with no constraints on the output values.

**Definition 2** (*Geometric Semantic Mutation*). Given a parent function $T : \mathbb{R}^n \rightarrow \mathbb{R}$, the geometric semantic mutation with mutation step $ms$ returns the real function $T_M = T + ms \cdot (T_{R1} - T_{R2})$, where $T_{R1}$ and $T_{R2}$ are random real functions.

Reference (Moraglio et al., 2012) formally proves that this operator corresponds to a box mutation on the semantic space, and induces a unimodal fitness landscape. However, even though without a formal proof, it is not difficult to have an intuition of it, considering that each element of the semantic vector of the offspring is a "weak" perturbation of the corresponding element in the parent's semantics. We informally define this perturbation as "weak" because it is given by a random expression centered in zero (the difference between two random trees). Nevertheless, by changing parameter $ms$, we are able to tune the "step" of the mutation, and thus the importance of this perturbation.

It is important to point out that at every step of one of these operators, offspring contain the complete structure of the parents, plus one or more random trees as its subtrees and some arithmetic operators: the size of each offspring is thus clearly much larger than the one of their parents. The exponential growth of the individuals in the population (demonstrated in Moraglio et al. (2012)) makes these operators unusable in practice: after a few generations the population becomes unmanageable because the fitness evaluation process becomes unbearably slow. The solution that is suggested in Moraglio et al. (2012) consists in performing an automatic simplification step after every generation in which the programs are replaced by (hopefully smaller) semantically equivalent ones. However, this additional step adds to the computational cost of GP and is only a partial solution to the progressive program size growth. Last but not least, according to the particular language used to code individuals and the used primitives, automatic simplification can be a very hard task. The work proposed in Vanneschi, Castelli, Manzoni, and Silva (2013) explained how the exponential growth of the individuals has been addressed, with a very simple and effective implementation of the GP algorithm. This is the implementation used in this paper.

## 4. Related works

Prediction of concrete strength is an important problem in the engineering field and several works have been proposed to address this problem. In this section a brief literature review about the use of computational intelligence techniques for facing this problem is proposed. In Saridemir, Topcu, Ozcan, and Severcan (2009), artificial neural networks and fuzzy logic models for prediction of long-term effects of ground granulated blast furnace slag on compressive strength of concrete under wet curing conditions have been developed. For purpose of constructing these models, 44 different mixes with 284 experimental data were gathered from the literature. Results have shown that artificial neural networks and fuzzy logic systems have strong potential for prediction of long-term effects of ground granulated blast furnace slag on compressive strength of concrete.

In Yeh (1998), author demonstrates the possibilities of adapting artificial neural networks to predict the compressive strength of high-performance concrete. A set of trial batches of HPC was produced in the laboratory and demonstrated satisfactory experimental results. The study led to two main conclusions: (1) a strength model based on artificial neural networks is more accurate than a model based on regression analysis; and (2) it is convenient and easy to use artificial neural networks models for numerical experiments to review the effects of the proportions of each variable on the concrete mix.

The work proposed in Lai and Serra (1997) also predicted the strength of concrete by using neural networks. In particular, a model is developed, based on neurocomputing, for predicting, with sufficient approximation, the compressive strength of cement conglomerates. Some neural networks are constructed in which the different mix-design parameters of a variety of cement conglomerates i.e. the compressive strength, are associated. Satisfactory results were obtained for evaluating the mechanical properties of different concrete mixes.

The work proposed in Lee (2003) used single and multiple artificial neural networks architecture for predicting of concrete strength, while authors of Dias and Pooliyadda (2001) utilized back propagation neural networks to predict the strength and slump of ready mixed concrete and high strength concrete in which chemical admixtures and/or mineral additives were used.

Authors of Akkurt, Ozdemir, Tayfur, and Akyol (2003) utilized genetic algorithms and ANN for modeling of compressive strength of cement mortar. In the study, the common three layer feed-forward type of artificial neural networks is used.

Other works that used artificial neural networks to model concrete strength are reported in Yeh,Sebastiá et al. (2003), Sebastiá, Olmo, and Irabien (2002) and Ren and Zhao (2002).

While artificial neural network represents the most studied technique in order to address the prediction of concrete strength other machine learning techniques have been investigated. For instance, in Demir (2005) the theory of fuzzy sets, especially fuzzy modeling, is discussed to determine elastic modulus of both normal and high-strength concrete. A fuzzy logic algorithm has been devised for estimating elastic modulus from compressive strength of concrete. The main advantage of fuzzy models is their ability to describe knowledge in a descriptive human-like manner in the form of simple rules using linguistic variables only. The use of fuzzy logic to predict cement strength is also proposed in Gao (1997).

While the use of Genetic Programming to predict material strength has not been widely investigated, some works have been proposed. The work in Gandomi, Alavi, and Sahab (2010) proposes a new approach for the formulation of compressive strength of carbon fiber reinforced plastic (CFRP) confined concrete cylinders using a promising variant of Genetic Programming namely, Linear Genetic Programming (LGP). The LGP-based models are constructed using two different sets of input data. The first set of inputs comprises diameter of concrete cylinder, unconfined concrete strength, tensile strength of CFRP laminate and total thickness of utilized CFRP layers. The second set includes unconfined concrete strength and ultimate confinement pressure which are the most widely used parameters in the CFRP confinement existing models. The results demonstrate that the LGP-based formulas are able to predict the ultimate compressive strength of concrete cylinders with an acceptable level of accuracy.

The work proposed in Chen (2003) is aimed at addressing a mixture-proportioning problem, which uses the macroevolutionary algorithm (MA) combined with Genetic Programming to estimate the compressive strength of HPC. Genetic Programming provides system identification in a transparent and structured way, while MA could improve the capability of searching global optima and avoid premature convergence during the selection process of GP. Experimental results show that the proposed model is better than the traditional proportional selection Genetic Programming for HPC strength estimation.



Q1 Please cite this article in press as: Castelli, M., et al. Prediction of high performance concrete strength using Genetic Programming with geometric semantic genetic operators. *Expert Systems with Applications* (2013), http://dx.doi.org/10.1016/j.eswa.2013.06.037

## 5. Experimental phase

### 5.1. Data set information

Following the same procedure described in Yeh (1998), experimental data from 17 different sources was used to check the reliability of the strength model. Data were assembled for concrete containing cement plus fly ash, blast furnace slag, and superplasticizer. A determination was made to ensure that these mixtures were a fairly representative group governing all of the major parameters that influence the strength of HPC and present the complete information required for such an evaluation. The dataset consist in 1028 instances, each of them described by 8 variables that are reported in Table 1.

### 5.2. Experimental settings

We tested the proposed implementation of GP with geometric semantic operators (GS-GP from now on) against a standard GP system (ST-GP). A total of 50 runs were performed with each technique: this is a fundamental aspect given the stochastic nature of the considered systems. In each run a different partition between training and test data has been considered. In particular 70% of the instances have been used as training data, while the remaining have been used as test data.

All the runs used a population of 200 individuals and the evolution stopped after 2000 generations. Trees initialization was performed with the Ramped Half-and-Half method (Koza, 1992) with a maximum initial depth equal to 6. The function set contained the four binary arithmetic operators +, −, ∗, and / pro-

**Table 2**
Experimental comparison between different Machine Learning techniques. Median error on training and test data over 50 runs has been reported for each technique.

| Method | Train | Test |
|---|---|---|
| Linear regression (Weisberg, 2005) | 10.567 | 10.007 |
| Square Regression (Seber & Wild, 2003) | 17.245 | 15.913 |
| Isotonic Regression (Hoffmann, 2009) | 13.497 | 13.387 |
| Radial Basis Function Network (Haykin, 1999) | 16.778 | 16.094 |
| SVM Polynomial Kernel (1 degree) (Ikopf & Smola, 2002) | 10.853 | 10.260 |
| SVM Polynomial Kernel (2 degree) | 7.830 | 7.614 |
| SVM Polynomial Kernel (3 degree) | 6.323 | 6.796 |
| SVM Polynomial Kernel (4 degree) | 5.567 | 6.664 |
| SVM Polynomial Kernel (5 degree) | 4.938 | 6.792 |
| Artificial Neural Networks (Haykin, 1999) | 7.396 | 7.512 |
| Standard GP | 7.792 | 8.67 |
| Geometric Semantic GP | 3.897 | 5.926 |

**Table 1**
Variables used to describe each instance. For each variable minimum, maximum, average, median and standard deviation values have been reported.

| | Minimum | Maximum | Average | Median | Standard Deviation |
|---|---|---|---|---|---|
| Cement ($kg/m^3$) | 102.0 | 540.0 | 281.2 | 272.9 | 104.5 |
| Fly ash ($kg/m^3$) | 0.0 | 359.4 | 73.9 | 22.0 | 86.3 |
| Blast furnace slag ($kg/m^3$) | 0.0 | 200.1 | 54.2 | 0.0 | 64.0 |
| Water ($kg/m^3$) | 121.8 | 247.0 | 181.6 | 185.0 | 21.4 |
| Superplasticizer ($kg/m^3$) | 0.0 | 32.2 | 6.2 | 6.4 | 6.0 |
| Coarse aggregate ($kg/m^3$) | 801.0 | 1145.0 | 972.9 | 968.0 | 77.8 |
| Fine aggregate ($kg/m^3$) | 594.0 | 992.6 | 773.6 | 779.5 | 80.2 |
| Age of testing (days) | 1.0 | 365.0 | 45.7 | 28.0 | 63.2 |

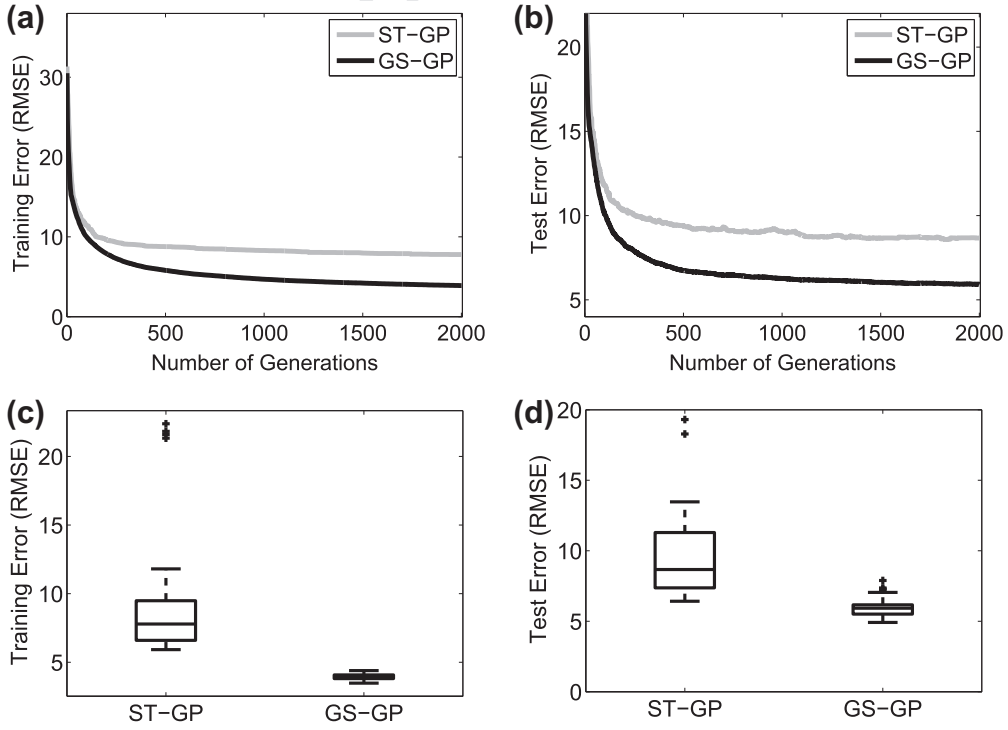


**Fig. 4.** Evolution of (a) training and (b) test errors for each technique, median of 50 runs. Boxplots of (c) training and (d) test fitness.

Q1 Please cite this article in press as: Castelli, M., et al. Prediction of high performance concrete strength using Genetic Programming with geometric semantic genetic operators. *Expert Systems with Applications* (2013), http://dx.doi.org/10.1016/j.eswa.2013.06.037

**Table 3**
*p*-values given by the statistical test.

| | | LIN | SQ | ISO | RBF | SVM-1 |
|---|---|---|---|---|---|---|
| GS-GP | TRAIN | $6.88 \cdot 10^{-18}$ | $6.88 \cdot 10^{-18}$ | $6.88 \cdot 10^{-18}$ | $6.88 \cdot 10^{-18}$ | $6.88 \cdot 10^{-18}$ |
| | TEST | $6.88 \cdot 10^{-18}$ | $6.88 \cdot 10^{-18}$ | $6.88 \cdot 10^{-18}$ | $6.88 \cdot 10^{-18}$ | $6.88 \cdot 10^{-18}$ |
| | | SVM-2 | SVM-3 | SVM-4 | SVM-4 | ANNs |
| GS-GP | TRAIN | $6.88 \cdot 10^{-18}$ | $6.88 \cdot 10^{-18}$ | $6.88 \cdot 10^{-18}$ | $6.88 \cdot 10^{-18}$ | $6.88 \cdot 10^{-18}$ |
| | TEST | $2.21 \cdot 10^{-16}$ | $3.04 \cdot 10^{-11}$ | $1.75 \cdot 10^{-7}$ | $1.40 \cdot 10^{-7}$ | $2.12 \cdot 10^{-15}$ |

tected as in Koza (1992). Fitness was calculated as the root mean square error (RMSE) between predicted and target values.

The terminal set contained 8 variables, each one corresponding to a different feature in the dataset. To create new individuals, ST-GP used standard (subtree swapping) crossover (Koza, 1992) and (subtree) mutation (Koza, 1992) with probabilities equal to 0.9 and 0.1 respectively. For GS-GP, crossover rate is 0.7, while mutation rate is 0.3 (with a mutation step of 1). The motivation for this different mutation (and crossover) rate for the two GP systems is that a preliminary experimental study has been performed (independently for the two systems) for finding the parameter setting able to return the best results. Survival from one generation to the other was always guaranteed to the best individual of the population (elitism). No maximum tree depth limit has been imposed during the evolution.

In the next section, experimental results are reported using curves of RMSE on the training and test set. In particular, at each generation the best individual in the population (i.e. the one that has the smaller training error) has been chosen and the value of its error on the training and test data has been stored. The reported curves finally contain the median of all these values collected at each generation. The median was preferred over the mean in the reported plots because of its higher robustness to outliers.

### 5.3. Results

Plots in Fig. 4(a) and (b) report training and test error at each generation, for ST-GP and GS-GP respectively and clearly show that GS-GP outperforms ST-GP on both training and test sets.

Plots in Fig. 4(c) and (d) report a statistical study of the results achieved by ST-GP and GS-GP, considering the 50 runs that have been performed. Denoting by *IQR* the interquartile range, the ends of the whiskers represent the lowest datum still within 1.5· *IQR* of the lower quartile, and the highest datum still within 1.5· *IQR* of the upper quartile. As it is possible to see, GS-GP outperforms ST-GP both on training and out of samples data. To analyze the statistical significance of these results, a set of tests has been performed on the median errors. As a first step, the Kolmogorov–Smirnov test has shown that the data are not normally distributed and hence a rank-based statistic has been used. Successively, the Wilcoxon rank-sum test for pairwise data comparison has been used under the alternative hypothesis that the samples do not have equal medians. The *p*-value obtained was $7.07 \cdot 10^{-18}$ when results on training data have been considered, while we obtained a *p*-value of $3.79 \cdot 10^{-16}$ when we considered results on test data. Therefore, when using the usual significance level $\alpha = 0.01$ we can clearly state that GS-GP produces solutions that are significantly lower (i.e., better) than the ones produced by ST-GP both on training and test data.

### 5.4. Comparison with other machine learning techniques

Besides comparing GS-GP with ST-GP, we are also interested in comparing GS-GP with other well-known state of the art ML methods to have an idea of the competitiveness of the obtained results.

To perform the comparison with other ML methods we used the implementations provided by the Weka public domain software (Weka Machine Learning Project). As done for the previous experimental phase, a preliminary study has been performed in order to find the best tuning of the parameters for the considered techniques.

The results of the comparison we performed are reported in Table 2.

From this table it is possible to state that GS-GP performs better than all the considered ML methods on both training and test instances. Beside GS-GP, SVM with polynomial kernel of fourth degree is the technique that produces the best performance on unseen data among the considered ones. Increasing the degree of the polynomial used by SVM's kernel, results in overfitting the training data. To assess if the differences between GS-GP and the consider ML techniques are statistical significant, we performed a Wilcoxon rank-sum test with a Bonferroni correction for the value of $\alpha$. The *p*-values are summarized in Table 3. From the results of the statistical test, we can state that GS-GP produces significantly better results on both training and test data with respect to all the other considered ML techniques.

## 6. Conclusions

High-performance concrete is a highly complex material that makes modeling its behavior a difficult task. In this study an intelligent GP-based system to predict the compressive strength of HPC has been proposed. The system uses recently defined geometric semantic genetic operators that present several advantages with respect to standard GP operators. In particular, they have the extremely interesting property of inducing a unimodal fitness landscape for any problem consisting in matching input data into known output ones (regression and classifications are instances of this general problem).

Experimental results show the suitability of the proposed system for the prediction of the compressive strength of HPC. In particular, the proposed method outperforms standard Genetic Programming and returns results that are significantly better to the ones produced by other well-known machine learning techniques.

## Acknowledgments

This work was supported by national funds through FCT under contract Pest-OE/EEI/LA0021/2011 and projects EnviGP (PTDC/EIA-CCO/103363/2008) and MassGP (PTDC/EEI-CTP/2975/2012). Portugal.

## References

Lomborg, B. (2001). *The skeptical environmentalist: Measuring the real state of the world*. Cambridge University Press.
Yeh, I.-C. (1998). Modeling of strength of high-performance concrete using artificial neural networks. *Cement and Concrete Research, 28*(12), 1797–1808.
Kumar, M., Singh, S. K., & Singh, N. (2012). Heat evolution during the hydration of portland cement in the presence of fly ash, calcium hydroxide and super plasticizer. *Thermochimica Acta, 548*(0), 27–32.

Abrams, D. A. (1927). Water-cement ration as a basis of concrete quality. *ACI Materials Journal, 23*(2), 452–457.
Nagaraj, T., & Banu, Z. (1996). Generalization of Abrams' law. *Cement and Concrete Research, 26*(6), 933–942.
Popovics, S. (1990). Analysis of concrete strength versus water-cement ratio relationship. *ACI Materials Journal, 87*(5), 517–529.
Bhanja, S., & Sengupta, B. (2005). Influence of silica fume on the tensile strength of concrete. *Cement and Concrete Research, 35*(4), 743–747.
Poli, R., Langdon, W. B., & Mcphee, N. F. (2008). A field guide to genetic programming. <http://www.gp-field-guide.org.uk>.
Koza, J. R. (1992). *Genetic programming: On the programming of computers by means of natural selection*. Cambridge, MA, USA: MIT Press.
Gusel, L., & Brezocnik, M. (2011). Application of genetic programming for modelling of material characteristics. *Expert Systems with Applications, 38*(12), 15014–15019.
Tsakonas, A., Dounias, G., Doumpos, M., & Zopounidis, C. (2006). Bankruptcy prediction with neural logic networks by means of grammar-guided genetic programming. *Expert Systems with Applications, 30*(3), 449–461.
Aslam, M. W., Zhu, Z., & Nandi, A. K. (2013). Feature generation using genetic programming with comparative partner selection for diabetes classification. *Expert Systems with Applications, 40*(13), 5402–5412.
Guo, L., Rivero, D., Dorado, J., Munteanu, C. R., & Pazos, A. (2011). Automatic feature extraction using genetic programming: An application to epileptic {EEG} classification. *Expert Systems with Applications, 38*(8), 10425–10436.
Kayadelen, C. (2011). Soil liquefaction modeling by genetic expression programming and neuro-fuzzy. *Expert Systems with Applications, 38*(4), 4080–4087.
Koza, J. R. (2010). Human-competitive results produced by genetic programming. *Genetic Programming and Evolvable Machines, 11*(3–4), 251–284.
Beadle, L., & Johnson, C. (2008). Semantically driven crossover in genetic programming. In *Proc. of the IEEE World Congress on Comput. Intelligence* (pp. 111–116). IEEE Press.
Beadle, L., & Johnson, C. G. (2009). Semantically driven mutation in genetic programming. In *2009 IEEE Congress on Evolutionary Computation, IEEE Computational Intelligence Society* (pp. 1336–1342). Trondheim, Norway: IEEE Press.
Krawiec, K., & Lichocki, P. (2009). Approximating geometric crossover in semantic space. In *GECCO '09* (pp. 987–994). ACM.
Jackson, D. (2010). Promoting phenotypic diversity in genetic programming. In *PPSN 2011th International Conference on Parallel Problem Solving From Nature. Lecture Notes in Computer Science* (Vol. 6239, pp. 472–481). Krakow, Poland: Springer.
Castelli, M., Vanneschi, L., & Silva, S. Semantic search based genetic programming and the effect of introns deletion. *IEEE Transactions on Human-Machine Systems*. http://dx.doi.org/10.1109/TSMCC.2013.2247754.
Moraglio, A., Krawiec, K., & Johnson, C. G. (2012). Geometric semantic genetic programming. In *Parallel Problem Solving from Nature PPSN XII (part 1). Lecture Notes in Computer Science* (Vol. 7491, pp. 21–31). Springer.
Stadler, P., & Institute, S. (1995). Towards a theory of landscapes. In *Complex Systems and Binary Networks. Lecture Notes in Physics* (Vol. 461–461, pp. 78–163). Berlin/Heidelberg: Springer.
Vanneschi, L. (2004) Theory and practice for efficient genetic programming, Ph.D. Thesis, Faculty of Sciences, University of Lausanne, Switzerland.
Altenberg, L. (1994). The evolution of evolvability in genetic programming (pp. 47–74). MIT Press (Ch. 3).
Vanneschi, L., Castelli, M., Manzoni, L., & Silva, S. (2013). A new implementation of geometric semantic gp and its application to problems in pharmacokinetics. In *EuroGP. Lecture Notes in Computer Science* (Vol. 7831, pp. 205–216). Springer.
Saridemir, M., Topcu, I. B., Ozcan, F., & Severcan, M. H. (2009). Prediction of long-term effects of GGBFS on compressive strength of concrete by artificial neural networks and fuzzy logic. *Construction and Building Materials, 23*(3), 1279–1286.
Lai, S., & Serra, M. (1997). Concrete strength prediction by means of neural network. *Construction and Building Materials, 11*(2), 93–98.
Lee, S.-C. (2003). Prediction of concrete strength using artificial neural networks. *Engineering Structures, 25*(7), 849–857.
Dias, W., & Pooliyadda, S. (2001). Neural networks for predicting properties of concretes with admixtures. *Construction and Building Materials, 15*(7), 371–379.
Akkurt, S., Ozdemir, S., Tayfur, G., & Akyol, B. (2003). The use of gaanns in the modelling of compressive strength of cement mortar. *Cement and Concrete Research, 33*(7), 973–979.
Yeh, I.-C. Design of high-performance concrete mixture using neural networks and nonlinear programming, *Journal of Computing in Civil Engineering* (Vol. 13). Q3
Sebastiá, M., Olmo, I. F., & Irabien, A. (2003). Neural network prediction of unconfined compressive strength of coal fly ash cement mixtures. *Cement and Concrete Research, 33*(8), 1137–1146.
Kim, J., & Kim, D. (2002). Application of neural networks for estimation of concrete strength. *KSCE Journal of Civil Engineering, 6*(4), 429–438.
Ren, L., & Zhao, Z. (2002). An optimal neural network and concrete strength modeling. *Advances in Engineering Software, 33*(3), 117–130.
Demir, F. (2005). A new way of prediction elastic modulus of normal and high strength concrete fuzzy logic. *Cement and Concrete Research, 35*(8), 1531–1538.
Gao, F.-L. (1997). A new way of predicting cement strength fuzzy logic. *Cement and Concrete Research, 27*(6), 883–888.
Gandomi, A., Alavi, A., & Sahab, M. (2010). New formulation for compressive strength of cfrp confined concrete cylinders using linear genetic programming. *Materials and Structures, 43*(7), 963–983.
Chen, L. (2003). Study of applying macroevolutionary genetic programming to concrete strength estimation. *Journal of Computing in Civil Engineering, 17*(4), 290–294.
Weka machine learning project. Weka. <http://www.cs.waikato.ac.nz/~ml/weka>.
Weisberg, S. (2005). *Applied linear regression. Wiley series in probability and statistics*. Wiley.
Seber, G., & Wild, C. (2003). *Nonlinear regression. Wiley series in probability and statistics*. Wiley.
Hoffmann, L. (2009). Multivariate isotonic regression and its algorithms. Wichita State University, College of Liberal Arts and Sciences, Department of Mathematics and Statistics.
Haykin, S. (1999). *Neural networks: A comprehensive foundation*. Prentice Hall.
Ikopf, B., & Smola, A. (2002). *Learning with kernels: Support vector machines, regularization, optimization and beyond. Adaptative computation and machine learning series*. The MIT Press.



Q1 Please cite this article in press as: Castelli, M., et al. Prediction of high performance concrete strength using Genetic Programming with geometric semantic genetic operators. *Expert Systems with Applications* (2013), http://dx.doi.org/10.1016/j.eswa.2013.06.037