Computers and Concrete, *Vol. 19, No. 6 (2017) 651-658*
DOI: https://doi.org/10.12989/cac.2017.19.6.651

An evolutionary system for the prediction of high performance concrete strength based on semantic genetic programming

Mauro Castelli[1a], Leonardo Trujillo[2b], Ivo Gonçalves[1,3c] and Aleš Popovič*[1,4]

[1]*NOVA IMS, Universidade Nova de Lisboa, 1070-312, Lisbon, Portugal*
[2]*Tree-Lab, Instituto Tecnológico de Tijuana, Tijuana B.C., 22500, México*
[3]*Department of Informatics Engineering, CISUC, University of Coimbra, 3030-290, Coimbra, Portugal*
[4]*Faculty of Economics, University of Ljubljana, Kardeljeva Ploščad 17, 1000, Ljubljana, Slovenia*

(Received January 16, 2016, Revised February 14, 2017, Accepted February 15, 2017)

Abstract. High-performance concrete, besides aggregate, cement, and water, incorporates supplementary cementitious materials, such as fly ash and blast furnace slag, and chemical admixture, such as superplasticizer. Hence, it is a highly complex material and modeling its behavior represents a difficult task. This paper presents an evolutionary system for the prediction of high performance concrete strength. The proposed framework blends a recently developed version of genetic programming with a local search method. The resulting system enables us to build a model that produces an accurate estimation of the considered parameter.

Experimental results show the suitability of the proposed system for the prediction of concrete strength. The proposed method produces a lower error with respect to the state-of-the art technique. The paper provides two contributions: from the point of view of the high performance concrete strength prediction, a system able to outperform existing state-of-the-art techniques is defined; from the machine learning perspective, this case study shows that including a local searcher in the geometric semantic genetic programming system can speed up the convergence of the search process.

Keywords: high performance concrete; concrete strength; genetic programming; local search; semantics

1. Introduction

Concrete is the most used material in civil engineering. Engineers must deal with the fact that in some structures, such as tunnels, nuclear plants, underground car parking, and high buildings ordinary concrete would not suffice. In these applications, high performance concrete (HPC) is commonly utilized (Viet-Thien-An *et al.* 2014). In addition to the basic ingredients used 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 *et al.* 2012, Mosabepranah and Eren 2016).

As pointed out in Aïtcin (2003), the development of HPC technology has shown what Féret expressed in 1892 in the original formula for estimating the compressive strength of a concrete mixture: concrete compressive strength is closely related to the compactness of the hardened matrix. Nevertheless, HPC is such a highly complex material that modeling its behavior is a difficult task.

The Abrams' water-to-cement ratio (w/c) law (Abram 1927, Nagaraj and Banu 1996) has been described as the most useful and significant advancement in the evolution of concrete technology. According to Abrams' law, in concrete materials for a mixture of workable consistency, the strength of concrete is determined by the ratio of water to cement (w/c). As the water content increases, the strength decreases. The implication of the Abrams' law, therefore, is that the strengths of various concrete are identical as long as their w/c ratios remain the same, regardless of the details of the compositions. A second implication is that only the quality of the cement paste controls the strength of comparable cement, while the paste quantity does not matter.

When the water-cement ratio law was proposed by Abrams, the use of fly ash and silica fume as replacements or substitutes for part of the cement was virtually unknown. Consequently, the effects of fly ash and silica fume were not considered in the development of Abrams' law (Oluokun 1994). With the development of HPC materials, since the early 1960s, concrete mix compositions have changed, and cement is no longer the only cementitious material in concrete mixes. In a high percentage of situations, today's cementitious material content is made up of cement plus fly ash.

An analysis of a variety of experimental data aimed at investigating the applicability of Abrams' law to concrete mixes containing fly ash, led to the conclusion that Abrams' water-cement ratio law is not directly applicable to mixes

∗Corresponding author, Professor
E-mail: ales.popovic@ef.uni-lj.si
[a]Professor
E-mail: mcastelli@novaims.unl.pt
[b]Professor
E-mail: leonardo.trujillo@tectijuana.edu.mx
[c]Professor
E-mail: igoncalves@novaims.unl.pt

ISSN: 1598-8198 (Print), 1598-818X (Online)

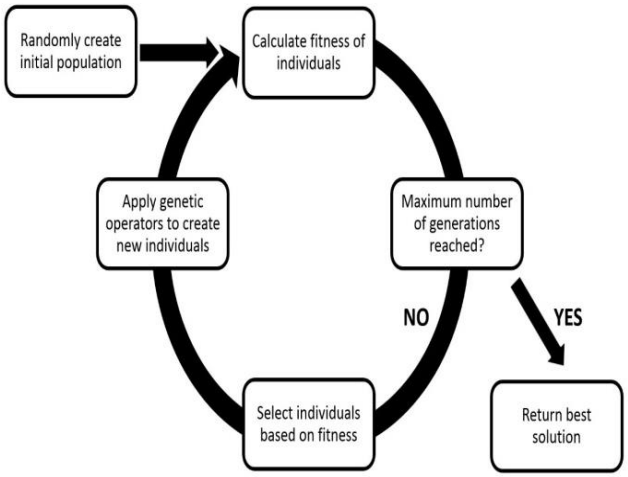

Fig. 1 The GP algorithm

with fly ash. As reported in Yeh (2008), several studies have independently 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 (Khan *et al.* 2016). 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).

Therefore, although experimental data have shown the practical acceptability of the Abrams' law within wide limits, the validity of this rule for concrete with supplementary cementitious materials (fly ash, blast furnace slag, etc.) should be investigated (Bhanja and Sengupta 2005, Ramadoss and Nagamani 2012). The more we know about the concrete composition versus strength relationship, the better our understanding of the nature of concrete and how to optimize the concrete mixture.

All these aspects highlight the need for reliable and accurate techniques that allow modeling the behavior of concrete materials. Machine learning (ML) methods have demonstrated their suitability in modeling the behavior of concrete materials. Among the different ML techniques, genetic programming (Koza 1992) has been used in different works, showing its suitability in modeling concrete characteristics. In Cevik and Sonebi (2008), a genetic programming system to model the performance of self-compacting SIFCON of cement slurries has been proposed, while Peng *et al.* (2009) used GP to model the strength of high-performance concrete. Other ML techniques have been used to model concrete behavior: Marti-Vargas *et al.* (2013) and Yeh (2008) used neural networks, while Parichatprecha and Nimityongskul (2009) considered a hybrid that combines genetic algorithms and neural networks. A fuzzy system was used in Ramezanianpour *et al.* (2009) for diagnosis assessment of reinforced concrete bridge decks.

Building upon previous research from Castelli *et al.* (2013b), we present a system based on a recently defined variant of genetic programming. More in detail, after introducing the concept of semantics in the field of genetic programming, we couple the semantics-based genetic programming system with a local search optimizer to increase the accuracy of the prediction of the HPC concrete strength.

The remainder of this paper proceeds as follows. Section 2 provides a general overview of genetic programming. Section 3 focuses on the variant of genetic programming employed in this work and describes the method followed to include a local searcher in the evolutionary search. Section 4 presents the data used in this work to evaluate the performance of the system, the experimental settings and the results achieved by the proposed framework. Finally, Section 5 concludes the paper summarizing the main findings of this work.

2. Genetic programming

Genetic Programming (GP) is one of the techniques that belongs to a larger computational intelligence research area called evolutionary computation. Computational intelligence is a set of nature-inspired computational methodologies and approaches to address complex real-world problems to which traditional approaches, first principles modeling or explicit statistical modeling, are ineffective or infeasible.

GP consists of the automated learning of computer programs by means of a process inspired by biological evolution (Koza 1992). Generation by generation, GP stochastically transforms a population of programs into a new, hopefully improved, population. The quality of a solution is expressed by using an objective function (also called fitness function). The search process of GP is graphically depicted in Fig. 1.

Hence, the recipe for solving a problem with GP is the following:

• Choose a representation space in which candidate solutions can be specified. This consists of choosing the primitives of the programming language that will be used to construct programs. A program is built up from a terminal set (the input variables of the problem and, optionally, a set of constant values) and a function set (the primitive operators).

• Specify a fitness measure for evaluating the quality of a solution. This involves the execution of a candidate solution on a suite of test cases. In the case of supervised learning, a distance-based function is employed to quantify the divergence of a candidate's behavior from the desired one.

• Define a parent selection and replacement policy. Central to every EA (Evolutionary Algorithm) is the concept of fitness-driven selection in order to exert an evolutionary pressure towards promising areas of the search space. The replacement policy determines the way in which newly created offspring programs replace their parents in the population.

• Design a variation mechanism to generate 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, and replacement is employed to improve the

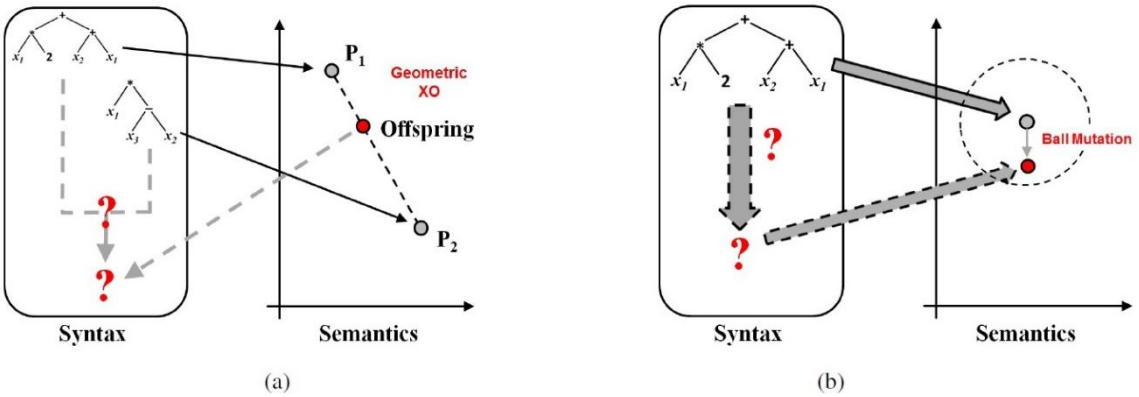

Fig. 2 Geometric semantic crossover (plot (a)) (and respectively geometric semantic mutation (plot (b))) performs a transformation on the syntax of the individual that corresponds to geometric crossover (respectively geometric mutation) in the semantic space

programs quality. This can be seen as a stepwise refinement.

To transform a population into a new population of candidate solutions, GP makes use of particular search operators called genetic operators. Considering the common tree representation of GP individuals, the standard genetic operators (crossover and mutation) act on the structure of the trees that represent the candidate solutions. In other terms, standard genetic operators act on the syntax of the programs. In this paper, we used genetic operators that, differently from the standard ones, are able to directly act at the semantic level. The definition of semantics used in this work is the one also considered in Moraglio *et al.* (2012) and will be presented in the following section.

To better clarify the differences between the genetic operators used in this work and the ones used in the standard GP algorithm, the latter are also briefly recalled. The standard 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 in detail, after choosing two individuals based on their fitness, the crossover operator performs the following operations: (1) selects a random subtree in each parent and (2) swaps the selected subtrees between the two parents. The resulting individuals are referred to as the offspring. The mutation operator introduces random changes in the structures of the individuals in the population. The most well-known mutation operator, called sub-tree mutation, works as follows: (1) it randomly selects a node in a tree, (2) It removes the node and the subtree for which it is the root, and (3) it inserts a randomly generated tree there. 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.

For a complete introduction to genetic programming, the reader is referred to the work of Koza (1992).

3. Methods

This section describes the components of the proposed computational intelligence system used for the prediction of high performance concrete strength. Section 3.1 describes the geometric semantic operators and their properties, while Section 3.2 presents the local search strategy that we used with the geometric semantic mutation. For a complete introduction to geometric semantic operators, the reader is referred to the work of Vanneschi (2017).

3.1 Geometric semantic operators

Despite the large number of human-competitive results achieved with the use of GP (Koza 2010), researchers still continue to develop new methods that improve the ability of GP to produce high-quality solutions. In recent years, one of the emerging ideas is to include the concept of semantics in the evolutionary process performed by GP. In this work we use the most common and widely accepted definition of semantics in GP literature (Krawiec and Lichocki (2009)). The semantics of a program T_i is defined as the vector of outputs $s_i=[T_i(x_1);T_i(x_2);\ldots;T_i(x_n)]$, obtained after executing the program on a set of data (Moraglio *et al.* 2012), where x_1, $x_2,\ldots,x_n$ are vectors containing the features of the problem. When T_i represents a real-valued function then $s_i \in R^n$.

In this section, we briefly recall the definition of the geometric semantic operators proposed by Moraglio *et al.* (2012). The objective of geometric semantic operators is to define modifications on the syntax of GP individuals that have a precise effect on their semantics. These operators define transformations in the syntax of GP individuals that correspond to well-known operators of Genetic Algorithms (GAs). In this way, GP can "inherit" the known properties of those GA operators. Furthermore, in the application of GP to supervised learning problems, the target point in the semantic space is also known (it corresponds to the vector of expected output values in supervised learning) and the fitness of an individual is simply given by the distance between the points it represents in the semantic space and the target point t.

It was shown in Moraglio *et al.* (2012) that when fitness is defined in this way it induces a unimodal error surface. The real-valued GA operators that we want to "map" into the GP semantic space are the geometric crossover and the ball mutation. In real-valued GAs, geometric crossover produces an offspring that lies on the segment that joins the

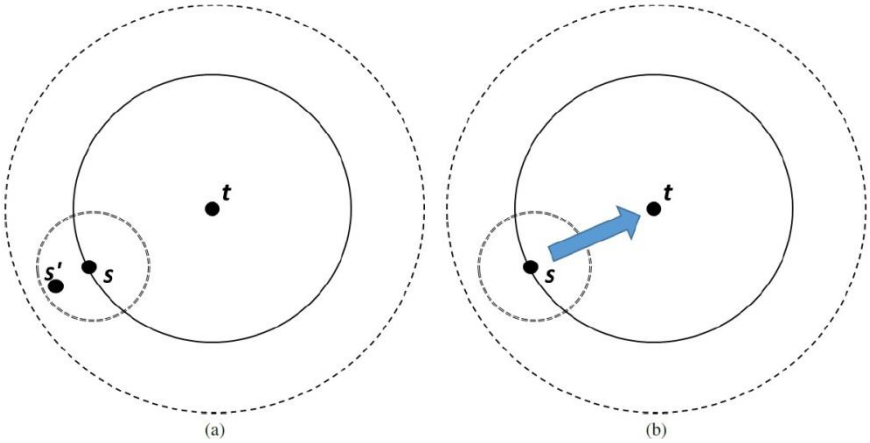

Fig. 3 A graphical representation of GSM(a) and GSM-LS(b)

parents. It was proven in Krawiec and Lichocki (2009) that in cases where the fitness is a direct function of the distance to the target (like the case we are interested in here) this offspring cannot have a worse fitness than the worst of its parents. Ball mutation consists of a random perturbation of the semantics of an individual.

Fig. 2 shows a graphical representation of the mapping between the syntactic and the semantic space produced by geometric semantic operators.

The definitions of the operators that correspond to geometric crossover and ball mutation in the GP semantic space, as given in Moraglio *et al.* (2012), are the following:

Geometric Semantic Crossover (GSC): given two parent functions T_1, T_2: $R^n \rightarrow R$, 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 function such that $T_R: R^n \rightarrow (0; 1)$.

To constrain T_R in producing values in (0; 1) we use the sigmoid function $T_R = \frac{1}{1+e^{-Trand}}$ where *Trand* is a random tree with no constraints on the output values.

Geometric Semantic Mutation (GSM): given a parent function T: $R^n \rightarrow R$, 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.

Moraglio *et al.* (2012) showed that the geometric semantic crossover corresponds to geometric crossover in semantic space (i.e., the point representing the offspring stands on the segment joining the points representing the parents) and the geometric semantic mutation corresponds to ball mutation on the semantic space (and thus induces a unimodal fitness landscape on the above mentioned types of problem). Moraglio *et al.* (2012) further showed that these operators create much larger offspring than their parents and the fast growth of the individuals in the population makes fitness evaluation unbearably slow, making the system unusable. Castelli *et al.* (2013a) and Castelli *et al.* (2014) proposed a possible solution to this problem, consisting of an implementation of Moraglio *et al.* (2012)'s operators that makes them not only usable in practice (Castelli *et al.* 2016a, Castelli *et al.* 2016b, Castelli *et al.* 2015c, Castelli *et al.* 2015d), but also very efficient. Their implementation is based on the idea that, besides storing the initial trees, at every generation it is enough to maintain in memory, for each individual, its semantics and a reference to its parents. Castelli *et al.* (2014) showed that the computational cost of evolving a population of n individuals for g generations is $O(ng)$. The cost of evaluating a new, unseen, instance is $O(ng)$ in the worst case, where all the individuals at each generation must be evaluated. Nonetheless, in practice, the average time needed to evaluate a new instance is $O(g)$ (Castelli *et al.* 2014). In the following sections, we will refer to the GP system that makes use of the geometric semantic operators as GSGP (Geometric Semantic Genetic Programming).

3.2 Local searcher in GSGP

This section presents the procedure we followed for including a local searcher in GSGP. The same approach has been described in Castelli *et al.* (2015a). In this work, we include a local searcher (LS) within the GSM operator, since previous works have shown that the mutation operator is the one that can benefit more from this modification (Vanneschi *et al.* 2014). It has also been recently shown that the GSM is the operator that more efficiently explores the search space in GSGP (Gonçalves *et al.* 2015). For this particular local searcher, the GSM with LS (GSM-LS) of a tree T generates an individual

$$T_{M=}\alpha_0 + \alpha_1 \cdot T + \alpha_2 \cdot (T_{R1} - T_{R2})$$

where $\alpha_i \in R$. Notice that α_2 replaces the mutation step parameter ms used in the definition of GSM. This in fact defines a basic multivariate linear regression problem, which can be solved, for example, by Ordinary Least Square regression (OLS). In this sense, after each mutation event, OLS is applied to the above expression to obtain the values of the model parameters (α_0, α_1, α_2) that best fit the training fitness cases.

Differently from existing works that relied on a non-linear local optimizer (Z-Flores *et al.* 2014), it is simple to apply a linear regression optimizer, given that the GSM operator defines a linear expression in the parameter space. By combining the exploration ability of GSGP with the exploitation ability of a local search method we expect to find good quality solutions in a small number of generations, hence avoiding the excessive specialization of

Table 1 Data considered in the experimental phase

	Minimum	Maximum	Mean	Median	Std. deviation
Cement (kg/m^3)	102	540	281.2	272.9	104.5
Fly ash (kg/m^3)	0	359.4	73.9	22	86.3
Blast furnace (kg/m^3)	0	200.1	54.2	0	64
Water (kg/m^3)	121.8	247	181.6	185	21.4
Superplasticizer (kg/m^3)	0	32.2	6.2	6.4	6
Coarse aggregate (kg/m^3)	801	1,145	972.9	968	77.8
Fine aggregate (kg/m^3)	594	992.6	773.6	779.5	80.2
Age of testing (days)	1	365	45.7	28	63.2

a model on the training instances and, consequently, overfitting.

To illustrate how GSM and GSM-LS differ, a graphical depiction of each method is provided in Fig. 3.

Fig. 3(a) shows a contour plot of semantic space, the space of all possible program outputs, with the highest fitness peak at the desired targets *t*. The semantics of a single GP tree is depicted as *s*; the circle around *s* is the area in which the semantics *s'* of the offspring generated by GSM may lie, where the radius of the circle is determined by the mutation step *ms*. Notice that GSM can, in some cases, generate offspring with semantics that are farther away from *t* than the parent, i.e., it can generate an offspring which is worse than its parent. This can slow down the convergence speed of the search. On the other hand, GSM-LS will always produce offspring that have a better fitness than the parent, by forcing the geometric mutation to always move in the direction of the known goal of the search, as depicted in Fig. 3(b).

The definition of the new GSM operator allows to counteract the problems that prevent GSGP to be used in applications where a vast amount of data is available by speeding up the convergence of the search. As demonstrated in the next section, GSGP with local search (LS-GSGP) can outperform GSGP by providing an accurate prediction of the HPC strength in a negligible amount of time.

4. Experimental phase

This section describes the experimental phase that has been performed to assess the performance of the proposed system. Section 4.1 describes the data considered, while section 4.2 presents the experimental settings and discusses the results achieved by the proposed system. Finally, section 4.3 compares the results achieved by the proposed system against the ones produced by non-evolutionary methods.

4.1 Data

The dataset used in the experimental phase was the same as in Castelli *et al.* (2013b). A description of the data can be found in that work. Here, we briefly summarize the features of the dataset. Data were assembled for concrete containing

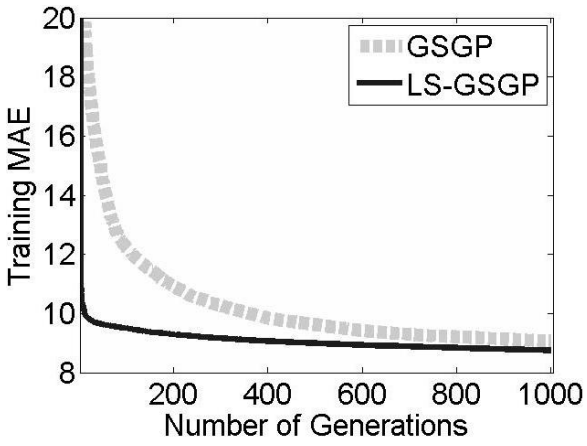

Fig. 4 Training MAE. The plot shows the median over 50 independent runs

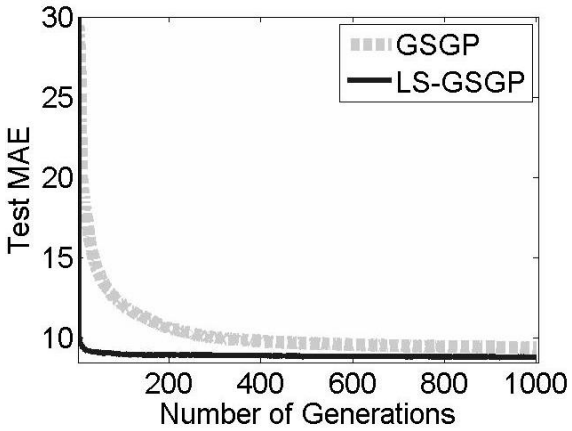

Fig. 5 Test MAE. The plot shows the median over 50 independent runs

cement plus fly ash, blast furnace slag, and superplasticizer. An analysis 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 consists of 1,028 instances, each of them described by 8 variables that are reported in Table 1.

4.2 Experimental settings and results

A comparison between the proposed semantics-based system with a local search optimizer (LS-GSGP) and GSGP was performed. GSGP has been selected for two main reasons: (1) as shown in Castelli *et al.* (2013b), GSGP is the state-of-the-art technique for addressing the HPC strength prediction problem, producing results that outperform other well-known machine learning techniques; (2) GSGP is the system that we want to improve by coupling the GSM operator with a local search optimizer.

Regarding the two GP systems, all the runs used populations of 250 individuals and evolution stops after 1000 generations. Tree initialization was performed with the Ramped Half-and-Half method described in Koza (1992), with a maximum initial depth equal to 6. The function set contained the arithmetic operators, including

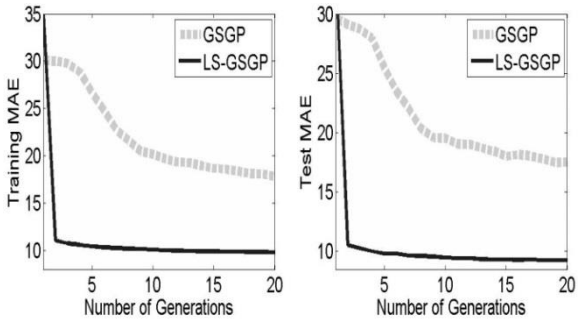

Fig. 6 First generations of the evolutionary process for training (left) and test fitness (right)

Table 2 Comparison between errors obtained on training and test instances by GSGP and LS-GSGP. For the considered systems we reported 10th, 25th, 50th, 75th and 90th percentile

	Training		Test	
	GSGP	LS-GSGP	GSGP	LS-GSGP
10th	8.7594	8.4792	8.6095	8.2239
25th	8.8627	8.5717	8.9824	8.4813
50th	9.0618	8.7537	9.4112	8.8226
75th	9.2399	8.9033	9.8345	9.1904
90th	9.4889	8.9989	10.2355	9.6026

the protected division as in Koza (1992). The terminal set contained 8 variables, each one corresponding to a different feature in the dataset. Mutation and crossover probabilities have been automatically self-tuned by using the system described in Castelli *et al.* (2015b). This system allows the practitioner to save the time-consuming task of tuning the GP parameters. Survival from one generation to the other was always guaranteed to the best individual of the population (elitism). For the GSM used in GSGP, a random mutation step has been considered in each mutation event, as suggested in Vanneschi *et al.* (2014). The results discussed in the next section have been obtained using the GSGP implementation (Vanneschi *et al.* 2013) freely available at http://gsgp.sourceforge.net and documented in Castelli *et al.* (2014).

We studied the performance by considering the Mean Absolute Error (MAE) as a measure of error. The definition of this error measure is the following

$$MAE = \frac{1}{N}\sum_{i \in Q} |t_i - y_i|$$

where y_i=$T(x_i)$ is the output of the GP individual T on the input data x_i and t_i is the target value for the instance x_i. N denotes the number of samples in the training or testing subset, and Q contains the indices of that set. Considering that the study aims at comparing the performance of different techniques, we prefer MAE over RMSE because it weights all the errors equally. Anyway, both error measures are commonly used in the machine learning area (Castelli *et al.* 2016a, Castelli *et al.* 2015d).

In this experimental phase we considered the median over 50 runs. We preferred the median with respect to the average for its robustness to outlier values. We start the discussion of the obtained results by considering the performance of GSGP and LS-GSGP. Results of this comparison are reported in Figs. 4 and 5.

Table 3 Comparison between errors obtained on training and test instances by LS-GSGP and other non-evolutionary methods. Median over 50 independent runs

Training		Test	
LS-GSGP	8.754	LS-GSGP	8.823
Square Regression	15.146	Square Regression	14.233
Radial Basis Function Network	14.790	Radial Basis Function Network	13.874
Isotonic Regression	11.637	Isotonic Regression	11.049
Abrams' Law	10.902	Abrams' Law	10.893

As it is possible to see from these plots, LS-GSGP outperforms GSGP on both training and test instances. Furthermore, besides the fitness values reached at the end of the search process, it is interesting to notice how LS-GSGP converges more quickly than GSGP. Hence, coupling the GSM operator with a local search optimizer allows LS-GSGP to converge in a smaller number of generations with respect to GSGP. This is fundamental for the applications where data are continuously provided and the training phase must be executed promptly. To better visualize the speed of convergence of LS-GSGP (compared with the one of GSGP), we report in Fig. 6 the first 20 generations of Figs. 4 and 5.

Regarding the numerical results (after 1000 generations), on the training instances LS-GSGP produces a MAE of 8.75, while GSGP produces a MAE of 9.06. On the test set the MAE obtained with LS-GSGP is 8.82, while GSGP produces a MAE of 9.41. Table 2 summarizes the MAE on training and test instances considering the 10th, 25th, 50th, 75th and 90th percentile.

To analyze the statistical significance of the obtained results, a set of tests has been performed on the median errors. Firstly, the Kolmogorov-Smirnov test has shown that data is not normally distributed (p-value smaller than $10E^{-10}$) and hence a rank-based statistic has been applied. Subsequently, the Mann-Whitney rank-sum test for pairwise data comparison has been used under the alternative hypothesis that the samples do not have equal medians. The p-values are 9.14E-09 for the training data and 3.80E-06 for the test data. Hence, LS-GSGP produces results that are statistically better than the ones produced by GSGP on both training and test instances.

To summarize, it is possible to state that the contribution proposed in this work to improve the performance of GSGP is effective. In particular, LS-GSGP can converge faster than GSGP and, more important, it is able to produce better overall results.

4.3 Comparison with other methods

This section reports the results obtained with other non-evolutionary techniques as well as the ones obtained with the application of the Abrams' law (strength at 28 days).

The objective of this experimental phase is to compare the performance of LS-GSGP against the ones achieved by other commonly used approaches. To perform the comparison with other methods, we used the implementation provided by the Weka public domain software (Hall *et al.* 2009). As done for the previous experimental phase, a preliminary study has been performed to find the best tuning of the parameters for the considered techniques. In particular, we used the functions provided by WEKA for finding the best parameter settings for the techniques taken into account: the tuning phase has been performed by using the WEKA metaclassifier (CVParameterSelection). The metaclassifier provides a way of automating the tuning process (Hall *et al.* 2009).

The techniques taken into account are the following: square regression (Seber and Wild 2003), radial basis function network (Haykin 1999) and isotonic regression (Hoffmann 2009). The results of the comparison we performed are reported in Table 3. The results show that LS-GSGP performs better than all the considered methods (and the difference among the MAE produced by LS-GSGP and the other methods is statistically significant) and, moreover, it produces a more accurate estimation with respect to the Abrams' law.

5. Conclusions

High-performance concrete is a highly complex material that makes modeling its behavior a challenging undertaking. Machine learning techniques have shown their suitability in addressing this problem and, among the several existing techniques, a variant of GP has demonstrated its superior performance. This GP system makes use of particular genetic operators that, differently from the standard genetic operators used in GP, work on the semantics of the solutions. While the use of semantics methods in GP has been successfully investigated, and applied, several important problems that do not allow to efficiently use these methods are still open. The GP system that uses the semantics operators (GSGP) requires a huge amount of generations in order to converge towards optimal solutions. This drawback affects the readability (i.e., the possibility, for a human being, of understanding the final model returned by the GP system) of the final model and requires a computational effort that is not adequate for real world applications.

To answer this call, this paper integrates a local search optimizer in the GSGP framework. By combining the exploration ability of GSGP with the exploitation ability of a local search method, we expect to find good quality solutions in a small number of generations. The proposed system, called LS-GSGP, can produce results that are statistically better than the ones produced by the GSGP algorithm that was reported in Castelli *et al.* (2013b). The system proposed in this paper represents the state-of-the-art method for addressing the HPC strength prediction problem. In particular, LS-GSGP is able to reduce the forecasting error with respect to GSGP, thus generating more accurate and reliable predictive models without overfitting the data. Finally, LS-GSGP is able to outperform several non-evolutionary techniques that are commonly used to address regression problems and, moreover, it returns a prediction that is more accurate with respect to the one provided by Abrams's law. We hope our work will spark future attempts in this intriguing research area.

References

Abrams, D.A. (1927), "Water-cement ration as a basis of concrete quality", *ACI Mater. J.*, **23**(2), 452-457.

Aïtcin, P.C. (2003), "The durability characteristics of high performance concrete: A review", *Cement Concrete Compos.*, **25**(4), 409-420.

Bhanja, S. and Sengupta, B. (2005), "Influence of silica fume on the tensile strength of concrete", *Cement Concrete Res.*, **35**(4), 743-747.

Castelli, A., Silva, S. and Vanneschi, L. (2014), "A C++ framework for geometric semantic genetic programming", *Gen. Program. Evol. Mach.*, **16**(1), 73-81.

Castelli, M., Castaldi, D., Giordani, I., Silva, S., Vanneschi, L., Archetti, F. and Maccagnola, D. (2013a), *An Efficient Implementation of Geometric Semantic Genetic Programming for Anticoagulation Level Prediction in Pharmacogenetics*, Progress in Artificial Intelligence, Volume 8154 of the series Lecture Notes in Computer Science, 78-89.

Castelli, M., Manzoni, L., Vanneschi, L., Silva, S. and Popovič, A. (2015b), "Self-tuning geometric semantic genetic programming", *Gen. Program. Evol. Mach.*, **17**(1), 55-74.

Castelli, M., Trujillo, L. Vanneschi, L., Silva, S., Z-Flores, E. and Legrand, P. (2015a), "Geometric semantic genetic programming with local search", *Proceedings of the 17th Annual Conference on Genetic and Evolutionary Computation*, Madrid, Spain, July.

Castelli, M., Trujillo, L., Vanneschi, L. and Popovič, A. (2015d), "Prediction of energy performance of residential buildings: A genetic programming approach", *Energy Build.*, **102**, 67-74.

Castelli, M., Trujillo, L., Vanneschi, L. and Popovič, A. (2016a), "Prediction of relative position of CT slices using a computational intelligence system", *Appl. Soft Comput.*, **46**, 537-542.

Castelli, M., Vanneschi, L. and De Felice, M. (2015c), "Forecasting short-term electricity consumption using a semantics-based genetic programming framework: The South Italy case", *Energy Econ.*, **47**, 37-41.

Castelli, M., Vanneschi, L. and Silva, S. (2013b), "Prediction of high performance concrete strength using genetic programming with geometric semantic genetic operators", *Exp. Syst. Appl.*, **40**(17), 6856-6862.

Castelli, M., Vanneschi, L., Manzoni, L. and Popovič, A. (2016b), "Semantic genetic programming for fast and accurate data knowledge discovery", *Swarm Evolut. Comput.*, **26**, 1-7.

Cevik, A. and Sonebi, M. (2008), "Modelling the performance of self-compacting SIFCON of cement slurries using genetic programming technique", *Comput. Concrete*, **5**(5), 475-490.

Gonçalves, I., Silva, S., Fonseca, C.M. (2015), "On the generalization ability of geometric semantic genetic programming", *Proceedings of the 18th European Conference on Genetic Programming*, March.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P. and Witten, I.H. (2009), "The WEKA data mining software: An update", *ACM SIGKDD Expl. Newslett.*, **11**(1), 10-18.

Haykin, S. (1999), *Neural Networks: A Comprehensive Foundation*, Prentice Hall.

Hoffmann, L. (2009), "Multivariate isotonic regression and its algorithms", Ph.D. Dissertation, Wichita State University, Kansas, U.S.A.

Khan, A., Do, J. and Kim, D. (2016), "Cost effective optimal mix

proportioning of high strength self compacting concrete using response surface methodology", *Comput. Concrete*, **17**(5), 629-638.

Koza, J.R. (1992), *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, MIT Press, Cambridge, U.S.A.

Koza, J.R. (2010), "Human-competitive results produced by genetic programming", *Gen. Program. Evol. Mach.*, **11**, 251-284.

Krawiec, K. and Lichocki, P. (2009), "Approximating geometric crossover in semantic space", *Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation*, Quebec, Canada, July.

Kumar, M., Singh, S.K. and Singh, N.P. (2012), "Heat evolution during the hydration of Portland cement in the presence of fly ash, calcium hydroxide and super plasticizer", *Thermochim. Acta*, **548**, 27-32.

Marti-Vargas, J.R., Ferri, F.J. and Yepes, V. (2013), "Prediction of the transfer length of prestressing strands with neural networks", *Comput. Concrete*, **12**(2), 187-209.

Moraglio, A., Krawiec, K. and Johnson, C.G. (2012), "Geometric semantic genetic programming", *Proceedings of the 12th International Conference on Parallel Problem Solving from Nature*, Volume 7491 of Lecture Notes in Computer Science, 21-31.

Mosabepranah, M.A. and Eren, O. (2016), "Statistical flexural toughness modeling of ultra-high performance concrete using response surface method", *Comput. Concrete*, **17**(4), 477-488.

Nagaraj, T. and Banu, Z. (1996), "Generalization of Abrams' law", *Cement Concrete Res.*, **26**(6), 933-942.

Oluokun, F.A. (1994), "Fly ash concrete mix design and the water-cement ratio law", *ACI Mater. J.*, **91**(4), 362-371.

Parichatprecha, R. and Nimityongskul, P. (2009), "An integrated approach for optimum design of HPC mix proportion using genetic algorithm and artificial neural networks", *Comput. Concrete*, **6**(3), 253-268.

Peng, C.H., Yeh, I. and Lien, L.C. (2009), "Modeling strength of high-performance concrete using genetic operation trees with pruning techniques", *Comput. Concrete*, **6**(3), 203-223.

Popovics, S. (1990), "Analysis of concrete strength versus water-cement ratio relationship", *ACI Mater. J.*, **87**(5), 517-529.

Ramadoss, P. and Nagamani, K. (2012), "Statistical methods of investigation on the compressive strength of high-performance steel fiber reinforced concrete", *Comput. Concrete*, **9**(2), 153-169.

Ramezanianpour, A.A., Shahhosseini, V. and Moodi, F. (2009), "A fuzzy expert system for diagnosis assessment of reinforced concrete bridge decks", *Comput. Concrete*, **6**(4), 281-303.

Seber, G. and Wild, C. (2003), *Nonlinear Regression*, Wiley Series in Probability and Statistics, Wiley.

Vanneschi, L. (2017), "An introduction to geometric semantic genetic programming", *Proceedings of the NEO 2015*, Tijuana, Mexico, September.

Vanneschi, L., Castelli, M., Manzoni, L. and Silva, S. (2013), "A new implementation of geometric semantic GP and its application to problems in pharmacokinetics", *Proceedings of the EuroGP 2013, European Conference on Genetic Programming*, 205-216.

Vanneschi, L., Silva, S., Castelli, M. and Manzoni, L. (2014), *Geometric Semantic Genetic Programming for Real Life Applications*, Genetic Programming Theory and Practice XI, 191-209.

Viet-Thien-An, V., Röbler, C., Bui, D. and Horst-Michael, L. (2014), "Rice husk ash as both pozzolanic admixture and internal curing agent in ultra-high performance concrete", *Cement Concrete Compos.*, **53**, 270-278.

Yeh, I. (1998), "Modeling of strength of high-performance concrete using artificial neural networks", *Cement Concrete Res.*, **28**(12), 1797-1808.

Yeh, I. (2008), "Modeling slump of concrete with fly ash and superplasticizer", *Comput. Concrete*, **5**(6), 559-572.

Z-Flores, E., Trujillo, L., Schütze, O. and Legrand, P. (2014), *Evaluating the Effects of Local Search in Genetic Programming*, EVOLVE-A Bridge between Probability, Set Oriented Numerics, and Evolutionary Computation, Volume 288 of the Series Advances in Intelligent Systems and Computing, 213-228.

CC