Combining Geometric Semantic GP with Gradient-descent Optimization

Gloria Pietropolli[1], Luca Manzoni[1], Alessia Paoletti[1], and Mauro Castelli[2]

[1] Dipartimento di Matematica e Geoscienze, Università degli Studi di Trieste
Via Alfonso Valerio 12/1, 34127 Trieste, Italy
gloria.pietropolli@phd.units.it, lmanzoni@units.it
[2] Nova Information Management School (NOVA IMS), Universidade Nova de Lisboa
Campus de Campolide, 1070-312, Lisboa, Portugal
mcastelli@novaims.unl.pt

Abstract. Geometric semantic genetic programming (GSGP) is a well-known variant of genetic programming (GP) where recombination and mutation operators have a clear semantic effect. Both kind of operators have randomly selected parameters that are not optimized by the search process. In this paper we combine GSGP with a well-known gradient-based optimizer, *Adam*, in order to leverage the ability of GP to operate structural changes of the individuals with the ability of gradient-based methods to optimize the parameters of a given structure.
Two methods, named HYB-GSGP and HeH-GSGP, are defined and compared with GSGP on a large set of regression problems, showing that the use of Adam can improve the performance on the test set. The idea of merging evolutionary computation and gradient-based optimization is a promising way of combining two methods with very different – and complementary – strengths.

1 Introduction

Genetic Programming (GP) [13] is one of the most prominent evolutionary computation techniques, with the ability to evolve programs, usually represented as trees, to solve specific problems given a collection of input and output pairs. Traditionally, operators in GP have focused on manipulating the syntax of GP individuals, like swapping subtrees for crossover or replacing subtrees for mutation. While simple to describe, these operations produce an effect on the *semantics* [25] of the individuals that can be complex to describe, with small variations in the syntax that may significantly affect the semantics. To address this problem, semantic operators were introduced. In particular, geometric semantic operators, first introduced in [15], have been used for defining Geometric Semantic GP (GSGP), a new kind of GP where crossover and mutation operators have a clear effect on the semantics. While in the original formulation GSGP was only of theoretical interest, due to the size of the generated individuals, the algorithm introduced in [24] provided a way for implementing GSGP efficiently.

While the introduction of GSGP helped in establishing a clear effect of recombination and mutation operators, also improving the quality of the generated

solutions, there is still a largely untapped opportunity of combining GSGP with local search methods. In particular, we can observe that, give two GP trees T_1 and T_2, their recombination is given by $\alpha T_1 + (1 - \alpha)T_2$, and the mutation of one of them is given by $T_1 + ms(R_1 - R_2)$, where R_1 and R_2 are two random trees. As we can observe, there are three parameters, α, $\beta = 1 - \alpha$, and ms that are either fixed or randomly selected during the evolution process. As long as each function used in the generation of the individuals is derivable, we can compute the *gradient* of the error with respect to the parameters used in crossover and mutation. Thus, we can employ a gradient-based optimizer to update the parameters of each crossover and mutation.

In this paper, we propose a way to combine GSGP and Adam, a well-known gradient-based optimizer. In some sense, by combining GSGP with a gradient-based optimizer, we are leveraging the strengths of each of the two methods: GSGP (and GP in general) is good at providing structural changes in the shape of the individuals, while gradient-based methods are perfect for optimizing a series of parameters of the individuals that the evolutionary process has difficulty in optimizing.

We experimentally show that the proposed method can provide better performance with respect to plain GSGP, thus suggesting that a combination of local search (via Adam) with GSGP is a new promising way to leverage knowledge from other areas of artificial intelligence: Adam and the gradient-based optimizers are well-studied in the area of neural networks, representing the main tool to perform the learning process in a neural network.

This paper is structured as follows: Section 2 provides an overview of the applications of local search to evolutionary methods and GP in particular. Section 3 recalls the reliant notions of GSGP (Section 3.1) and the Adam algorithm (Section 3.2). Section 3.3 introduces the proposed hybridized algorithms combining GSGP and the Adam algorithm. The experimental settings and the dataset used in the experimental validation are described in Section 4 and the results of the experimental campaign are presented in Section 5. Section 6 summarizes the main contributions of the paper and provides directions for further research.

2 Related Works

The combination of Evolutionary Algorithms (EAs) and local search strategies received greater attention in recent years [5,17,6]. While EAs can explore large areas of the search space, the evolutionary search process improves the programs in a discontinuous way [20]. On the other hand, when considering local optimizers, the solutions can be improved gradually and steadily in a continuous way. Thus, as stated by Z-Flores *et al.* [7], a hybrid approach that combines EAs with a local optimizer can result in a well-performing search strategy. Such approaches are a simple type of memetic search [5], and the basic idea is to include within the optimization process an additional search operator that, given an individual, searches for the local optima around it. Thanks to the possibility of fully exploiting the local region around each individual, memetic algorithms obtained excellent

results over different domains [17,5], and they outperform evolutionary algorithms in multimodal optimisation [18]. Despite these results, the literature presents a poor number of contributions dealing with GP [23], thus indicating that the GP community may have not addressed the topic adequately. Some examples are the works of Eskridge [8] and Wang [26] that are domain-specific memetic techniques not addressing the task of symbolic regression considered in this work. Muñoz *et al.* [16], proposed a memetic algorithm that, given a regression (or classification) problem, creates a new feature space that is subsequently considered for addressing the underlying optimization problem. The algorithm, by maximizing the mutual information [12] in the new feature space, shows superior results with respect to other state-of-the-art techniques.

Focusing on the use of gradient descent in GP, the existing contributions are focused on particular tasks or particular components of the solutions. For instance, Topcyy *et al.* [22] analyzed the effectiveness of gradient search optimization of numeric leaf values in GP. In particular, they tuned conventional random constants utilizing gradient descent, and they considered several symbolic regression problems to demonstrate the approach's effectiveness. Zhang *et al.* [27] applied a similar strategy to address object classification problems and, also in this case, better results were obtained compared to the ones achieved with standard GP. Graff *et al.* [9] employed resilient backpropagation with GP to address a complex real-world problem concerning wind speed forecasting, showing improved results. In [21], the authors used gradient-descent search to make partial changes of certain parts of genetic programs during evolution. To do that, they introduced weight parameters for each function node, what the authors call inclusion factors. These weights modulate the importance that each node has within the tree. The proposed method, which uses standard genetic operators and gradient descent applied to the inclusion factors, outperformed the basic GP approach that only uses standard genetic operators (i.e., without gradient descent and inclusion factors).

The aforementioned contributions are related to syntax-based GP. In the context of semantics-based GP [25], the integration of a local search strategy into GP was proposed by Castelli *et al.* [4] with the definition of a specific semantic mutation operator. Experimental results showed excellent performance on the training set, but with a severe overfitting [3]. To the best of our knowledge, this is the only attempt to integrate a local optimizer within semantic GP.

In this paper, we follow a different strategy, and, to create a hybrid semantic GP framework, we rely on gradient descent optimization.

3 Gradient Descent GSGP

This section will discuss the two tools that will be combined later in this work. First, Geometric Semantic GP is described. Later, Adam, one of the most powerful gradient descent optimizers, is introduced and discussed. Afterward, the main contribution of this paper, i.e., the first integration of a gradient descent optimizer within GSGP, is presented.

3.1 Geometric Semantic GP

Traditional genetic programming investigates the space of programs exploiting search operators that analyze their syntactic representation. To improve the performance of GP, recent years have witnessed the integration of semantic awareness in the evolutionary process [25]. The semantic of a solution can be identified by the vector of its output values calculated on the training data. Thus, we can represent a GP individual as a point in a real finite-dimensional vector space, the so-called semantic space. Geometric Semantic Genetic Programming (GSGP) is an evolutionary technique originating from GP that directly searches the semantic space of the programs. GSGP has been introduced by Moraglio and coauthors [15], together with the definition of the correspondent Geometric Semantic Operators (GSOs). These operators replace traditional (syntax-based) crossover and mutation, inducing geometric properties on the semantic space. GSOs induce on the training data a unimodal error surface for any supervised learning problem where input data has to match with known targets. More precisely: given two parents functions T_1, $T_2 : \mathbb{R}^n \to \mathbb{R}$, *Geometric Semantic Crossover* (GSC) generates the real function $T_{XO} = (T_1 \cdot T_R) + ((1 - T_R) \cdot T_2)$, where T_R is a random real functions whose output range in the interval $[0, 1]$. Similarly, given a parent function $T : \mathbb{R}^n \to \mathbb{R}$, *Geometric Semantic Mutation* (GSM) generates the real functions $T_M = T + ms \cdot (T_{R1} - T_{R2})$ where T_{R1} and T_{R2} are random real functions whose output range in the interval $[0, 1]$ and ms is a parameter called mutation step. This means that GSC generates one offspring whose semantics stands on the line joining the semantics of the two parents in the semantic space, while GSM generates an individual contained in the hyper-sphere of radius ms centred in the semantics of the parent in the semantic space. An intrinsic GSGP's problem is that this technique leads to larger offsprings with respect to their parents. Due to this issue, the algorithm becomes excessively unbearably slow generation after generation, making it unsuitable for real-world applications. In [24,2], Vanneschi and coauthors introduced a GSGP implementation that solves this problem and consists in storing only the semantic vectors of newly created individuals, besides storing all the individuals belonging to the initial population and all the random trees generated during the generations. This improvement turn the cost of evolving g generations of n individuals from $\mathbf{O}(ng)$ to $\mathbf{O}(g)$. The same idea was subsequently adopted to reconstruct the best individual found by GSGP, thus allowing for its usage in a production environment [1].

3.2 Adam algorithm

Adam (Adaptive Moment Estimation) [11] is an algorithm for first-order gradient-based optimization of stochastic objective functions, based on adaptive estimates of lower-order models. Adam optimizer is efficient, easy to implement, requires little memory usage for its execution, and is well suited for problems dealing with a vast amount of data and/or parameters. The steps performed by the Adam optimizer are summarized in Algorithm 1. The inputs required for this

method are the parametric function $f(\theta)$, the initial parameter vector θ_0, the number of steps N, the learning rate α, the exponential decay rate of the first momentum β_1, the one for the second momentum β_2, and ϵ, set by default at 10^{-8}. At every iteration, the algorithm updates first and second moment estimates using the gradient computed with respect to the stochastic function f. These estimates are then corrected to contrast the presence of an intrinsic initialization bias through the divisions described in line 7 and 8, where β_1^{i+1} stands for the element-wise exponentiation. For further details about the implementation of the Adam optimizer and the demonstration of its properties, the reader can refer to [11].

Algorithm 1 Pseudocode of the *Adam* algorithm.

Require: $f(\theta),\ \theta_0,\ N,\ \alpha,\ \beta_1 \in [0,1),\ \beta_2 \in [0,1),\ \epsilon$
1: $m_0 \leftarrow 0$
2: $v_0 \leftarrow 0$
3: **for** $i = 0 \cdots N$ **do**
4: $\quad d_{i+1} \leftarrow \nabla_\theta f_{i+1}(\theta_i)$
5: $\quad m_{i+1} \leftarrow \beta_1 \cdot m_i + (1-\beta_2) \cdot d_{i+1}$
6: $\quad v_{i+1} \leftarrow \beta_2 \cdot v_i + (1-\beta_2) \cdot d_{i+1}^2$
7: $\quad \bar{m}_{i+1} \leftarrow m_{i+1}/(1-\beta_1^{i+1})$
8: $\quad \bar{v}_{i+1} \leftarrow v_{i+1}/(1-\beta_2^{i+1})$
9: $\quad \theta_{i+1} \leftarrow \theta_i - \alpha \cdot \bar{m}_{i+1}/(\bar{v}_{i+1})$
10: **end for**

3.3 GSGP hybridized with Gradient Descent

The idea introduced in this work is to combine the strength of the two methods presented above, i.e., GSGP and the Adam optimizer. Geometric semantic GP, thanks to the geometric semantic operators, allows big jumps in the solution space. Thus, new areas of the solution space can be explored, with GSOs also preventing the algorithm to get stuck in a local optimum. Adam optimizer, on the other hand, is a gradient-based optimization technique. Thus, it performs small shifts in the local area of the solution space. A combination of these techniques should guarantee a jump in promising areas (i.e., where good-quality solutions lie) of the solution space, thanks to the evolutionary search of GSGP and subsequent refinement of the solution obtained with the Adam algorithm. Let's describe in more detail how to implement this combination. Let us consider an input vector $\boldsymbol{x}$ of n features, and the respective expected scalar value output y. By applying GSGP, an initial random population of functions in n variables is created. After performing the evolutionary steps involving GSM and GSC, a new population $T = (T_1, T_2, \cdots, T_N)$ of N individuals is obtained. The resulting vector T is composed of derivable functions, as they are obtained through additions, multiplications, and compositions of derivable functions. At this point, to understand for which parameter we should differentiate T, it is necessary to

introduce an equivalent definition of the geometric semantic operators presented in Section 3.1. In particular let us redefine the *Geometric Semantic Crossover* as $T_{XO} = (T_1 \cdot \alpha) + ((1 - \alpha) \cdot T_2)$, where $0 \leq \alpha \leq 1$, and the *Geometric Semantic Mutation* as $T_M = T + ms \cdot (R_1 - R_2)$, where $0 \leq m \leq 1$. As the values of α and m are randomly initialised, we can derive T with respect to α, $\beta = (1 - \alpha)$ and m. Therefore, the Adam optimizer algorithm can be applied, considering as objective function $f(\theta)$ the generation considered, while the parameter vector becomes $\theta = (\alpha, \beta, m)$. Thus, GSGP and Adam optimizer can be applied together to find the best solution for the problem at hand. We propose and investigate two ways to combine them:

- HYB-GSGP: the abbreviation stands for *Hybrid Geometric Semantic Genetic Programming*. Here, one step of GSGP is alternated to one step of the Adam optimizer.
- HeH-GSGP: the abbreviation stands for *Half et Half Geometric Semantic Genetic Programming*. Here, initially, all the GSGP genetic steps are performed, followed by an equal number of Adam optimizer steps.

In the continuation of the paper, we will refer to these two methods using the abbreviations just introduced.

4 Experimental Settings

This section describes the datasets considered for validating our technique (Section 4.1) and provides all the experimental settings (Section 4.2) to make the experiments completely reproducible. The code, for the complete reproducibility of the proposed experiments, is available at https://github.com/gpietrop/GSGP-GD [19].

4.1 Dataset

To assess the validity of the technique proposed in Section 3.3, real-world, complex datasets, ranging from different areas, have been considered and tested. All of them have been widely used as benchmarks for GP, and their properties have been discussed in [14]. Table 1 summarizes the characteristics of the different datasets, such as the number of instances and the number of variables. The objective of the first group of datasets is the prediction of pharmacokinetic parameters of potential new drugs. *Human oral bioavailability* (**%F**) measures the percentage of initial drug dose that effectively reaches the system blood circulation; *Median lethal dose* (**LD50**) measures the lethal dose of a toxin, radiation, or pathogen required to kill half the members of a tested population after a specified test duration; *Protein-plasma binding level* (**%PPB**) corresponds to the percentage of the initial drug dose that reaches the blood circulation and binds the proteins of plasma. Also, datasets originating from physical problems are considered: *Yacht hydrodynamics* (**yac**) measures the hydrodynamic performance of sailing yachts starting from its dimension and velocity; *Concrete slump* (**slump**) measures the

value about the slump flow of the concrete; *Concrete compressive strength* (**conc**) measures values about the compressive strength of concrete; *Airfoil self-noise* (**air**) is a NASA dataset obtained from a series of aerodynamic and acoustic test of airfoil blade sections.

Dataset	Variables	Instances	Area	Task
%F	242	359	Pharmacokinetic	Regression
LD50	627	234	Pharmacokinetic	Regression
%PPB	627	131	Pharmacokinetic	Regression
yac	7	308	Physics	Regression
slump	10	102	Physics	Regression
conc	9	1030	Physics	Regression
air	6	1503	Physics	Regression

Table 1. Principal characteristics of the considered datasets: the number of variables, the number of instances, the domain, and the task request.

4.2 Experimental Study

For all the datasets described in Section 4.1, samples have been split among train and test sets: 70% of randomly selected data has been used as a training set, while the remaining 30% has been used as a test set. For each dataset, 100 runs have been performed, each time with a random train/test split.

To assess the performance of HYB-GSGP and HeH-GSGP, the results obtained within these methods are compared to the ones achieved with classical GSGP. The comparison with the performance achieved by standard GP is not reported, because after some preliminary tests it has been observed that standard GP is non competitive against GSGP. We considered two hyperparameters settings to evaluate our methods' performance with different values assigned to the learning rate of the Adam algorithm. Both of them are compared against 200 generations of standard GSGP. To make the comparison fair, the total number of fitness evaluations must be equal for every method considered: 200 generations in the standard GSGP routine correspond to a combination of 100 generations of GSGP plus 100 steps of Adam optimizer, both for HYB-GSGP and HeH-GSGP.

The first learning rate value we considered is 0.1 and we will refer to HYB-GSGP and HeH-GSGP where Adam optimizer used this hyperparameter as, respectively, HYB-0.1 and HeH-0.1. The second learning rate value we considered is 0.01 and we will refer to HYB-GSGP and HeH-GSGP where Adam optimizer used this hyperparameter as, respectively, HYB-0.01 and HeH-0.01. The population size for all the considered systems is set to 50, and the trees of the first generation are initialized with the ramped half and half technique. Further details concerning the implementation of the semantic system and the Adam optimization algorithm are reported in Table 2. The considered fitness function is the Root Mean Squared Error (RMSE).

Parameter	Value
Function Set	$+, -, *, //$
Max. Initial Depth	6
Crossover Rate	0.9
Mutation Rate	0.3
Mutation step	0.1
Selection Method	Tournament of size 4
Elitism	Best individuals survive
Learing Rate - A (α)	0.1
Learing Rate - B (α)	0.01
Exponential Decay Rate - First Momentum (β_1)	0.9
Exponential Decay Rate - Second Momentum (β_2)	0.99
ϵ	10^{-8}

Table 2. Experimental settings. A horizontal line separates the parameters belonging to GSGP algorithm and the ones belonging to the Adam technique.

5 Experimental Results

		GSGP	HYB-0.1	HYB-0.01	HeH-0.1	HeH-0.01
%F	Train	38.08	37.74	**36.80**	39.61	40.60
	Test	40.15	40.48	**39.61**	40.85	41.23
LD50	Train	2118.00	**2086.56**	2128.22	2144.27	2161.00
	Test	2214.78	**2203.25**	2229.87	2221.72	2215.09
%PPB	Train	30.15	27.00	**24.32**	34.79	33.26
	Test	328.1	401.43	263.81	**213.86**	235.53
yac	Train	**11.83**	11.92	12.48	12.28	12.31
	Test	11.92	**11.83**	12.52	12.38	12.48
slump	Train	4.56	3.47	**2.92**	5.19	4.41
	Test	5.08	3.63	**3.32**	5.77	4.76
conc	Train	9.62	8.86	**8.50**	10.59	10.05
	Test	9.65	8.88	**8.69**	10.47	10.07
air	Train	27.76	31.54	**21.98**	30.37	30.46
	Test	27.94	31.71	**21.97**	30.15	30.53

Table 3. Training and testing fitness (RMSE) for the considered benchmark problems. Bold font indicates the best results.

As stated in Section 3.3, the goal of this study is to compare the performance of GSGP against the one obtained by the proposed methods.

For each problem, the median of the fitness (calculated over the 100 runs performed), for both the training and the validation sets, is displayed in Table 3. The corresponding statistical analysis is reported Figure 1 (for the test set), thorough letter-value plots. Letter-value plots are a particular kind of box-plots,

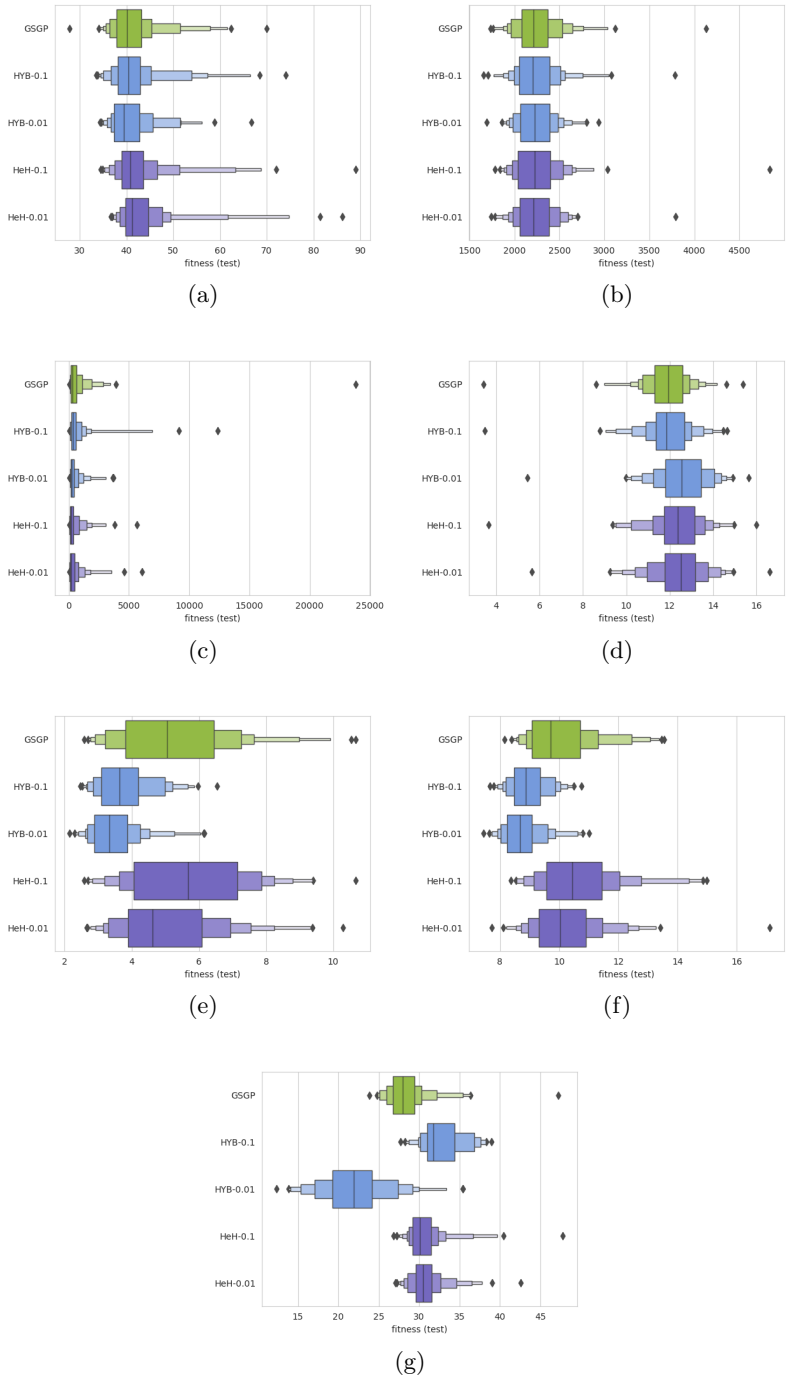

Fig. 1. Boxplots of Testing RMSE obtained over 100 independent runs of the considered benchmark problems. (a) %F, (b) LD50, (c) %PPB, (d) yac, (e) slump, (f) conc, (g) air.

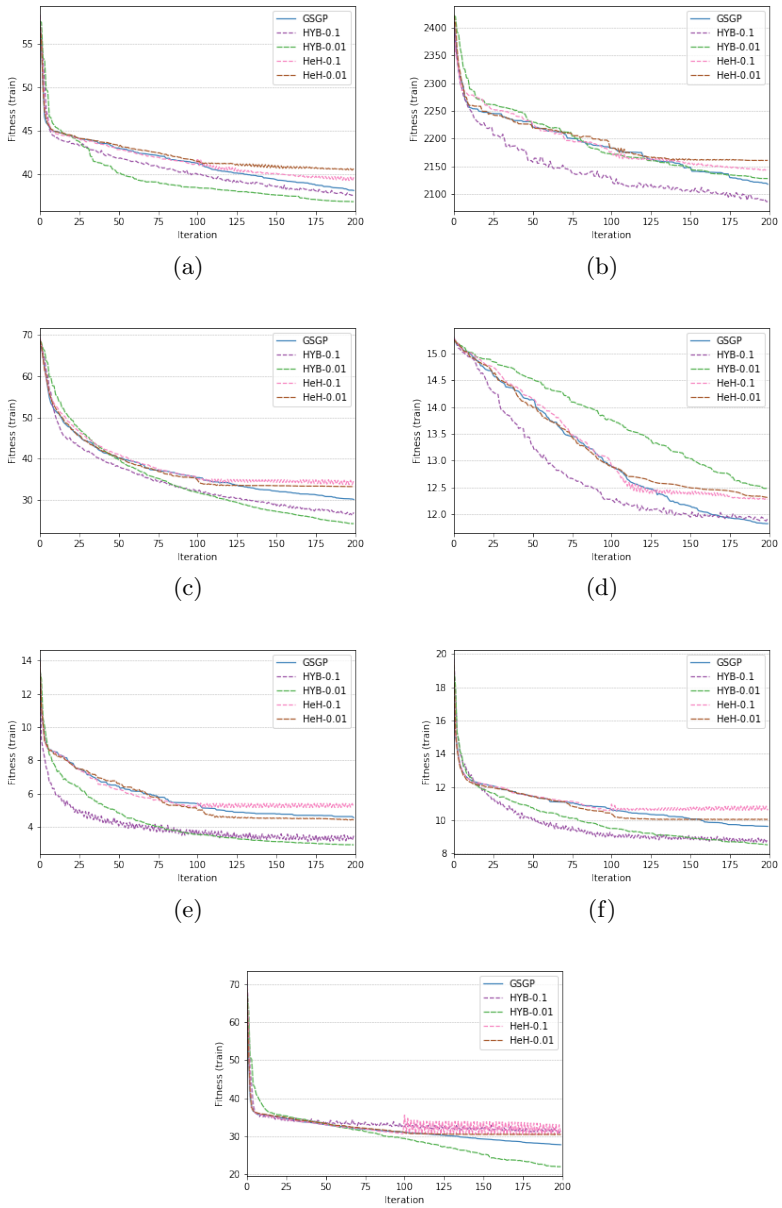

Fig. 2. Median of training fitness over 100 independent runs for the considered benchmark problems. (a) %F, (b) LD50, (c) %PPB, (d) yac, (e) slump, (f) conc, (g) air.

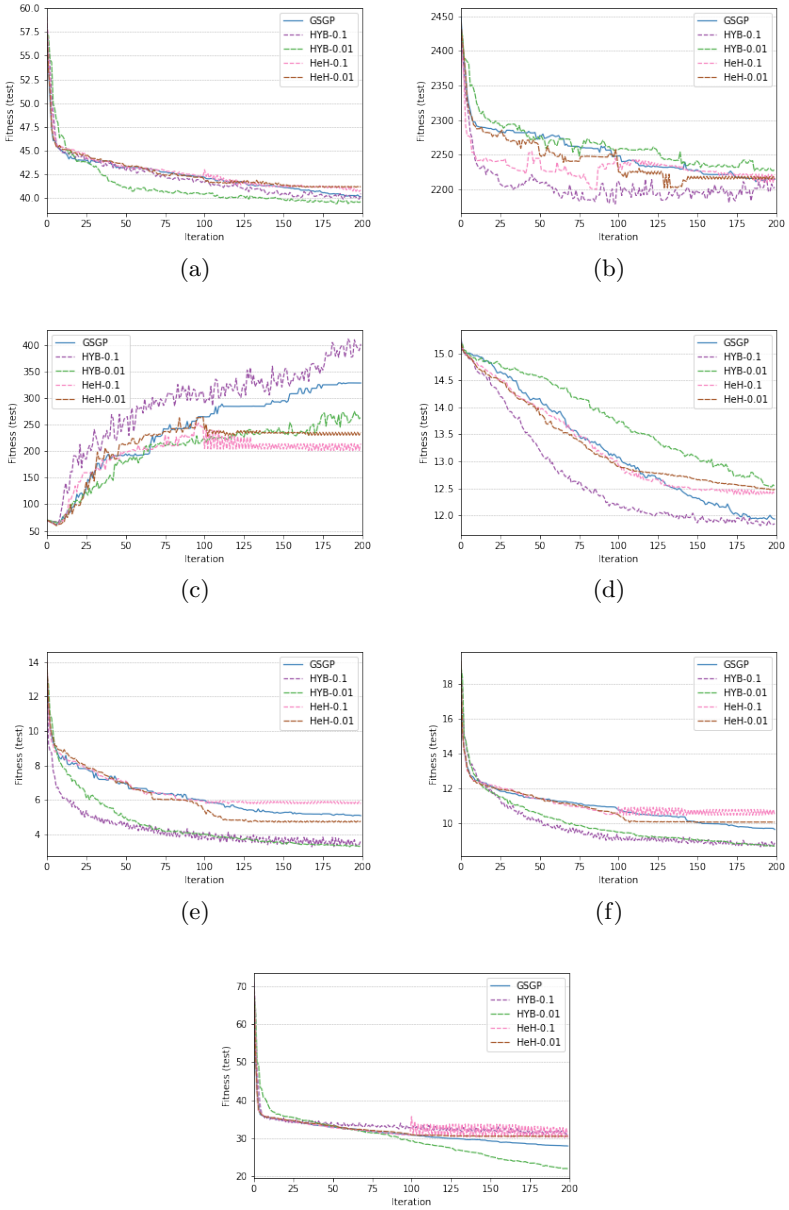

Fig. 3. Median of testing fitness over 100 independent runs for the considered benchmark problems. (a) %F, (b) LD50, (c) %PPB, (d) yac, (e) slump, (f) conc, (g) air.

introduced for the first time in [10]. We preferred them, over traditional box-plots, because they provide information not only about the distribution of the data but also about the tail behavior beyond the quartiles. Finally, the median fitness, at each generation, for the training and validation set, is displayed, respectively, in Figure 2 and in Figure 3. Training results are reported for the sake of completeness. Anyway, to compare the performance of the proposed methods against GSGP, we focus our analysis on the results achieved on the test set.
Table 3 shows that HYB-GSGP outperforms standard geometric semantic GP, while, most of the time, the HeH-GSGP method produces higher errors with respect to the two competitors.

		%F	LD50	%PPB	*yac*	*slump*	*conc*	*air*
HYB-0.1	Train	0.01484	0.0	0.0	0.9512	0.0	0.0	1.0
	Test	0.5978	0.4143	0.6716	0.9512	0.0	0.0	1.0
HYB-0.01	Train	0.5978	0.3820	0.0	0.8948	0.0	0.0	0.0
	Test	0.0631	0.6862	0.2158	1.0	0.0	0.0	0.0
HeH-0.1	Train	0.9998	1.0	1.0	1.0	0.9998	0.9998	1.0
	Test	0.8384	0.5445	0.0018	1.0	0.9989	0.9992	1.0
HeH-0.01	Train	1.0	0.9984	1.0	1.0	0.4805	0.9573	1.0
	Test	1.0	0.6923	0.0074	1.0	0.4805	0.9652	1.0

Table 4. P-values returned by the Wilcoxon ran-sum test under the alternative hypothesis that the median errors obtained from classical GSGP are smaller or equal than the median errors of other methods considered, i.e. HYB-0.1, HYB-0.01, HeH-0.1, HeH-0.01

Considering the %F dataset (Figure 1(a)), it is possible to see that the best results are achieved with the HYB-GSGP method in which the learning rate of the Adam algorithm is 0.01. This performance improvement is achieved after 25 epochs and is maintained throughout all the epochs performed, as shown in Figure 3(a).
With respect to the LD50 problem (Figure 1(b)), HYB-GSGP outperforms standard GSGP. In this case, the performance improvement is achieved with a learning rate of 0.1. However, on this benchmark, all the considered techniques perform similarly, with the fitness values on the test set that do not differ significantly.

Concerning the %PPB dataset (Figure 1(c)), it is clear that all the models are affected by overfitting. Hence, our expectation would suggest that lower error on the training set should lead to higher error on the test set. However, the HYB-GSGP method is able to perform better than GSGP, both in the training and validation set. Thus, HYB-GSGP seems to be capable of (slightly) reducing overfitting.

Taking into account the *yac* problem (Figure 1(d)), the best results on the test set are obtained with HYB-0.1 and, again, this method reaches such a performance improvement in approximately 25 epochs. As shown in Figure 3(d),

the test fitness of GSGP decreases linearly, while for HYB-0.1 fitness decreases more rapidly, leading the hybrid method to a faster converge.

For the *slump* dataset (Figure 1(e)), the combination of GSGP and Adam optimizer is successful, as HYB-GSGP outperforms standard GSGP for both the considered learning rate values. Moreover, also HeH-0.01 achieved better fitness values with respect to GSGP. Also in this case, it is interesting to highlight that the performance improvement provided by HYB-GSGP is achieved in a few epochs (Figure 3(e)). A similar behaviour can be observed for the *conc* dataset (Figure 1(f)): HYB-GSGP outperforms classical GSGP after a few epochs. Concerning the *air* problem (Figure 1(g)), the HYB-GSGP method with the Adam's learning rate of 0.01 leads to a significant improvement in terms of test fitness. On the other hand, the other hybrid methods introduced in this work are characterized by some instability.

Table 4 reports a statistical significance assessment of the result achieved. In particular, Table 4 displays the p-values obtained from the Wilcoxon rank-sum test for pairwise data comparison, with $\alpha = 0.05$, applied under the alternative hypothesis that the median errors resulting from the considered techniques are smaller or equal than the median errors obtained with classical GSGP. The statistical tests show that, on the test set, the HYB method, in particular HYB-0.01, obtains better results than GSGP on three benchmark problems.

While there is no clear superiority of one method with respect to the others, it is interesting to note, especially in Fig. 1, how the distribution of the results obtained by the hybrid methods is usually as tight or tighter than the distribution produced by GSGP, showing consistent results that are, in some cases, better than GSGP. Thus, it appears that using half of the "evolutionary" iterations coupled with local search via gradient-descent optimization can actually improve the results.

6 Conclusions

This paper investigates the possibility of integrating a gradient-based optimization method, the Adam algorithm, within a genetic programming system, GSGP. The idea behind this work relies on the possibilities of exploiting and combining the advantages of these two methods to achieve faster convergence of the evolutionary search process.

Two different ways of combining these methods have been investigated. In the former, denoted as HYB-GSGP, a step of GSGP is alternated to a step of Adam. In the latter, denoted as HeH-GSGP, first, all the GSGP steps are performed, and, subsequently, the refinement with Adam is executed. The results achieved with these two methods were compared against classical GSGP on eight real-world complex benchmark problems belonging to different applicative domains. The experiments were performed considering two different values of the learning rate, which is the most relevant parameter of the Adam algorithm.

Experimental results show that, in each of the considered benchmarks, HYB-GSGP outperforms classic GSGP in both training and test sets (with a statistically

significant difference on the test set on three problems). These results corroborate our hypothesis: the combination of GSGP with the Adam optimizer can improve the performance of GSGP. Moreover, HYB-GSGP converges to good-quality solutions faster than classical GSGP. In more detail, HYB-GSGP requires fewer epochs to converge, and the performance achieved by GSGP at the end of the evolutionary process is worse than the one achieved by the proposed hybrid method after a few fitness evaluations. On the contrary, the HeH-GSGP does not outperform GSGP even if it generally ensures good quality results on the test set. Thus, the results suggest that a combination of one step of GSGP and one step of Adam is the best way to mix these techniques.

This work represents the first attempt to create a hybrid framework between GSGP and a gradient descent-based optimizer. Considering the promising results obtained in this first analysis, this work paves the way to multiple possible future developments focused on improving the benefits provided by this kind of combination.

References

1. Castelli, M., Manzoni, L.: GSGP-C++ 2.0: A geometric semantic genetic programming framework. SoftwareX **10**, 100313 (2019)
2. Castelli, M., Silva, S., Vanneschi, L.: A C++ framework for geometric semantic genetic programming. Genetic Programming and Evolvable Machines **16**(1), 73–81 (2015)
3. Castelli, M., Trujillo, L., Vanneschi, L.: Energy consumption forecasting using semantic-based genetic programming with local search optimizer. Computational intelligence and neuroscience **2015** (2015)
4. Castelli, M., Trujillo, L., Vanneschi, L., Silva, S., Z-Flores, E., Legrand, P.: Geometric semantic genetic programming with local search. In: Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation. pp. 999–1006 (2015)
5. Chen, X., Ong, Y.S., Lim, M.H., Tan, K.C.: A multi-facet survey on memetic computation. IEEE Transactions on Evolutionary Computation **15**(5), 591–607 (2011)
6. Črepinšek, M., Liu, S.H., Mernik, M.: Exploration and exploitation in evolutionary algorithms: A survey. ACM computing surveys (CSUR) **45**(3), 1–33 (2013)
7. Emigdio, Z., Trujillo, L., Schütze, O., Legrand, P., et al.: Evaluating the effects of local search in genetic programming. In: EVOLVE-A Bridge between Probability, Set Oriented Numerics, and Evolutionary Computation V, pp. 213–228. Springer (2014)
8. Eskridge, B.E., Hougen, D.F.: Imitating success: A memetic crossover operator for genetic programming. In: Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No. 04TH8753). vol. 1, pp. 809–815. IEEE (2004)
9. Graff, M., Pena, R., Medina, A.: Wind speed forecasting using genetic programming. In: 2013 IEEE Congress on Evolutionary Computation. pp. 408–415. IEEE (2013)
10. Hofmann, H., Kafadar, K., Wickham, H.: Letter-value plots: Boxplots for large data. Tech. rep., had.co.nz (2011)
11. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization (2017)
12. Kojadinovic, I.: On the use of mutual information in data analysis: an overview. In: Proc Int Symp Appl Stochastic Models Data Anal. pp. 738–47 (2005)

13. Koza, J.R., Koza, J.R.: Genetic programming: on the programming of computers by means of natural selection, vol. 1. MIT press (1992)
14. McDermott, J., White, D.R., Luke, S., Manzoni, L., Castelli, M., Vanneschi, L., Jaskowski, W., Krawiec, K., Harper, R., De Jong, K., et al.: Genetic programming needs better benchmarks. In: Proceedings of the 14th annual conference on Genetic and evolutionary computation. pp. 791–798 (2012)
15. Moraglio, A., Krawiec, K., Johnson, C.G.: Geometric semantic genetic programming. In: International Conference on Parallel Problem Solving from Nature. pp. 21–31. Springer (2012)
16. Muñoz, L., Trujillo, L., Silva, S., Castelli, M., Vanneschi, L.: Evolving multidimensional transformations for symbolic regression with m3gp. Memetic Computing **11**(2), 111–126 (2019)
17. Neri, F., Cotta, C.: Memetic algorithms and memetic computing optimization: A literature review. Swarm and Evolutionary Computation **2**, 1–14 (2012)
18. Nguyen, P.T.H., Sudholt, D.: Memetic algorithms outperform evolutionary algorithms in multimodal optimisation. Artificial Intelligence **287**, 103345 (2020)
19. Pietropolli, G.: Gsgp-gd. https://github.com/gpietrop/GSGP-GD (2022)
20. Smart, W., Zhang, M.: Continuously evolving programs in genetic programming using gradient descent. Tech. Rep. CS-TR-04-10, Computer Science, Victoria University of Wellington, New Zealand (2004)
21. Smart, W., Zhang, M.: Continuously evolving programs in genetic programming using gradient descent. In: Proceedings of The Second Asian-Pacific Workshop on Genetic Programming, page 16pp, Cairns, Australia, 2004. (2004)
22. Topchy, A., Punch, W.F., et al.: Faster genetic programming based on local gradient search of numeric leaf values. In: Proceedings of the genetic and evolutionary computation conference (GECCO-2001). vol. 155162. Morgan Kaufmann (2001)
23. Trujillo, L., Emigdio, Z., Juárez-Smith, P.S., Legrand, P., Silva, S., Castelli, M., Vanneschi, L., Schütze, O., Muñoz, L., et al.: Local search is underused in genetic programming. In: Genetic Programming Theory and Practice XIV, pp. 119–137. Springer (2018)
24. Vanneschi, L., Castelli, M., Manzoni, L., Silva, S.: A new implementation of geometric semantic gp and its application to problems in pharmacokinetics. In: European Conference on Genetic Programming. pp. 205–216. Springer (2013)
25. Vanneschi, L., Castelli, M., Silva, S.: A survey of semantic methods in genetic programming. Genetic Programming and Evolvable Machines **15**(2), 195–214 (2014)
26. Wang, P., Tang, K., Tsang, E.P., Yao, X.: A memetic genetic programming with decision tree-based local search for classification problems. In: 2011 IEEE Congress of Evolutionary Computation (CEC). pp. 917–924. IEEE (2011)
27. Zhang, M., Smart, W.: Genetic programming with gradient descent search for multiclass object classification. In: European Conference on Genetic Programming. pp. 399–408. Springer (2004)