Towards the development of a complete GP system on an FPGA using geometric semantic operators

Carlos Goribar-Jimenez,
Yazmin Maldonado
and Leonardo Trujillo
Instituto Tecnológico de Tijuana,
Baja California, México
Email:
cgoribar@tectijuana.edu.mx,
yaz.maldonado@tectijuana.edu.mx,
leonardo.trujillo@tectijuana.edu.mx

Mauro Castelli,
Ivo Gonçalves,
and Leonardo Vanneschi
NOVA IMS, Universidade Nova de Lisboa,
1070-312 Lisboa, Portugal
Email:
mcastelli@novaims.unl.pt,
igoncalves@novaims.unl.pt,
lvanneschi@novaims.unl.pt

***Abstract*—Genetic Programming (GP) has been around for over two decades and has been used in a wide range of practical applications producing human competitive results in several domains. In this paper we present a discussion and a proposal of a GP algorithm that could be conveniently implemented on an embedded system, as part of a broader research project that pursues the implementation of a complete GP system in a Field Programmable Gate Array (FPGA). Motivated by the significant time savings associated with such a platform, as well as low power consumption, low maintenance requirements, small size of the system and the possibility of performing several parallel processes. The proposal is focused on the Geometric Semantic Genetic Programming (GSGP) approach that has been recently introduced with promising results. GSGP induces a unimodal fitness landscape, simplifying the search process. The experimental work considers five variants of GSGP, that incorporate local search strategies, optimal mutations and alignment in error space. Best results were obtained by a simple variant that uses both the optimal mutation step and the standard geometric semantic mutation, using three difficult real-world problems to evaluate the methods, outperforming the original GSGP formulation in terms of fitness and empirical convergence.**

I. Introduction

Genetic programming (GP) is a very flexible and general evolutionary paradigm, with several different variants proposed over the years [1]. For instance, regarding representation, trees are the most used data structure, but others have been proposed including linear representations [20], [21], graph representations [22] and stack-based approaches [25].

One recent and successful variant is the Geometric Semantic GP approach (GSGP), developed by Moraglio et al. [11]. The original GSGP formulation has shown to be quite good at handling difficult real world problems [14], [15], [18], [28] and to be resilient to overfitting [28]. Moreover, the form of the GSGP search operators induce a unimodal fitness landscape and also allows for the easy integration of greedy local optimizers or local search methods, leading to faster convergence, as well as smaller training and testing errors on several real-world tasks, outperforming standard GP [12], [13], [16], [17]. Another reason why GSGP is of interest, is the manner in which it can be efficiently implemented [16]. This implementation allows for the creation of a single set of initial trees, with the rest of the evolutionary process operating directly at the level of semantics, so there is no need to create, store or manipulate large trees of different shapes and sizes. This fact makes GSGP a promising alternative for the development of a GP algorithm that runs directly on-chip, for instance using an Field Programmable Gate Array (FPGA). The FPGA is a device considered to be at the intersection of software and hardware-oriented systems, its basic internal architecture consist of a huge two dimensional array of logic blocks interconnected by programmable switches. Although most modern FPGAs contain special embedded resources such as DSP units, memory blocks and microprocessors, the availability of these resources varies from one model to another, and from vendor to vendor, making it hard to write portable programs that run on any FPGA family. Our goal is to develop a highly portable system that can run on most FPGAs. Therefore, the best option to tackle this problem is to exclude special blocks and design the whole system by using only its array of logic blocks. However, this makes the design process much more complex, because even simple arithmetic blocks have now to be designed by hand. For example standard FPGAs do not support even basic operations such as division operations nor fixed/floating point format. For a deeper discution of implementations of GP on FPGAs see [26][27] in which a proposal of a random tree generator for FPGA is given, for arithmetic and floating point operations on FPGAs refer to [7].

While standard GSGP might be directly transferable to a device such as an FPGA, the methods that apply local searchers during the application of the genetic operators are not as amenable to be used within a hardware implementation [12], [13], [16], [17]. The main difficulty for the implementation of the local search methods in the FPGA is due to the considerable effort and time needed to develop hardware architectures in general, and particularly complex numerical methods like Singular Value Decomposition (SVD) or LU

978-1-5090-4601-0/17/$31.00

and QR factorization [29]. Therefore, choosing an appropriate GSGP implementation will be paramount for the development of our main research goal, that is the implementation of a GP algorithm in an embedded system.

In this work, we present, analyze and evaluate several recent GSGP variants, some of which are based on previous works, while others are proposed in this paper. In particular we consider:

- GSGP, as implemented in [16].
- GSGP-LS, GSGP with a local search process embedded in the Geometric Semantic Mutation [13].
- GSGP with optimal alignment in error space, where the goal is to find two aligned solutions that can be used to reconstruct the problem target [12].
- GSGP with an optimal mutation step; in this case, we modify the original formulation [17], such that the pseudo-inverse of the local optimization process is simplified allowing for an easier future implementation in hardware.
- Hybrid GSGP version, where the original geometric semantic operators are used in combination with the locally optimized variants.

Our results are encouraging, showing that a simple hybrid GSGP version outperforms all other GSGP variants studied here, using difficult real-world datasets. Moreover, the simplicity of the best performing variant makes it a very good choice for the further development of a fully hardware-based GSGP system on an FPGA chip.

The remainder of this paper is organized as follows. Section II provides a brief introduction to GSGP, particularly focusing on geometric semantic mutation. Then, Section III surveys several GSGP variants that exploit the geometric structure of semantic space to help improve algorithm convergence or performance; some of the discussed methods are taken from recent literature while others are original proposals of the current work. The experimental work is presented in Section IV, describing the problems used for evaluation, implementation details and discussing the main results. Finally, concluding comments are given in Section V along with an outline of future work, building towards a full FPGA implementation of GSGP algorithms.

II. Geometric Semantic Genetic Programming

Traditional GP utilizes operators that manipulate the syntactic representation of programs ignoring their effect on program output or semantics; i.e., the search is blind to the effects on fitness. In contrast, GSGP searches directly the space of the underlying semantics of programs by means of its special operators defined in [11]. The semantics of an individual or program P is defined as the program output, where, for each input vector $\overrightarrow{x_i}$ returns the scalar value $P(\overrightarrow{x_i})$. In other words, the *semantics* of P is $\overrightarrow{s_P} = [P(\overrightarrow{x_1}), P(\overrightarrow{x_2}), ..., P(\overrightarrow{x_n})]$.

1) Geometric Semantic Mutation (GSM).: In [11], the GSM was defined as a search operator $M : S \rightarrow S$ which is a *geometric ϵ-mutation* with respect to the metric d if for any chosen parent p, the offspring $o = M(p)$ is located in the metric ball of radius ϵ centered in the parent. In particular it is defined as

$$T_M = T + ms \cdot (T_{R1} - T_{R2}) \tag{1}$$

where T_{R1} and T_{R2} are random real functions, and ms is the mutation step.

For the remainder of this work, we will only consider GSM as the sole search operator in GSGP, given that it has been shown to be the best configuration in several real-world problems [14], [15], [18], [28]

III. GSGP variants

This section presents the variants of the GSGP algorithms that will be evaluated in this work.

A. GSGP with Local Search

One disadvantage of GSGP is that it tends to converge rather slowly and thus produces extremely large trees due to the accumulative effect of repeatedly applying the Geometric Semantic Operators (GSO's). One way to speed up the search process is found in [13] which proposes a hybrid approach that integrates a local search (LS). The local searcher is included in the GSM operator and is defined as

$$T_M = \alpha_0 + \alpha_1 \cdot T + \alpha_2 \cdot (T_{R1} - T_2) \tag{2}$$

where $\alpha_i \in \mathbb{R}$, this gives an overdetermined multivariate linear fitting problem solved using SVD. Authors argue that the operator is not a LS in the entire semantic space, rather, it should be seen as a LS operator that attempts to determine the best linear combination of the parent three and the random trees used to perturb it (R_1 and R_2).

B. GSGP with an Optimal Mutation Step (OMS)

Similar to the approach described above, [17] proposes a modification to the GSM operator using an optimal mutation step (OMS). The mutation operation in Eq. 1 is in fact a linear combination of the semantic vector of the parent $\overrightarrow{S_P}$ and a random semantic vector $\overrightarrow{RI}$ which is in turn the difference between the semantics of two random individual T_{R1} and T_{R2}. Given that $\overrightarrow{t}$ is the semantics of the desired target, the equation turns into

$$\overrightarrow{t} = \overrightarrow{S_P} + ms * \overrightarrow{RI} \tag{3}$$

then solving for ms,

$$ms = \left(\overrightarrow{t} - \overrightarrow{S_P}\right) * \overrightarrow{RI}^{-1} \tag{4}$$

where $\overrightarrow{RI}^{-1}$ is the pseudo-inverse of vector $\overrightarrow{RI}$.

The introduction of the adaptive mutation method achieves competitive generalization in only a single application of the mutation operators, consequently smaller solutions can be found with good generalization. In spite of the benefits of the adaptive mutation it is prone to quick overfitting [17]. Nonetheless, the OMS method is optimal in the sense of least squares (when using Moore-Penrose method) for each application of the operator [13], [17].

1) Proposed Optimal Mutation Step implementation: The OMS requires the computation of the pseudo-inverse of the $\overrightarrow{RI}$ vector. In [17] it is recommend to apply the Moore-Penrose pseudo-inverse which can be calculated via the Singular Value, LU or QR decomposition methods. However, those numerical methods require calculations that are too complex to be implemented in an embedded system like an FPGA [29]. As $\overrightarrow{RI}$ is always a vector of $m \times 1$ (being m the number of training cases), a general method is not required to compute the pseudoinverse for an $m \times 1$ matrix. Instead, the Moore-Penrose pseudo-inverse can be calculated for the special case when

$$A = \begin{pmatrix} a_1 \\ \vdots \\ a_m \end{pmatrix} \tag{5}$$

then, the pseudo-inverse A^+ of A is given by

$$A^+ = \frac{1}{\|A\|^2} A^* = \frac{1}{\|A\|^2} (\overline{a_1}, \cdots, \overline{a_m}) \tag{6}$$

where,

$$\|A\| = \sqrt{|a_1|^2 + \cdots + |a_m|^2}. \tag{7}$$

Considering all the elements of RI as real values then $\overline{a_i} = a_i$, and Eq. 6 can be rewritten as

$$A^+ = \frac{(a_1, \cdots, a_m)}{|a_1|^2 + \cdots + |a_m|^2} = \frac{A^T}{A^T \cdot A}. \tag{8}$$

Using Eq. 8 is easier and requires less computational effort than using a numerical method such as SVD. Thus, since Eq. 8 only requires arithmetic calculations and is a good candidate for an embedded implementation and it is the method we use in the experiments reported in this work.

C. GSGP with Error Alignment

The notion of the error vector was introduced in [12] as $\overrightarrow{e}_P = \overrightarrow{s}_P - \overrightarrow{t}$ that can be represented as a point in an n-dimensional space called *error space*. In other words, each vector in the semantic space is translated in the error space by subtracting $\overrightarrow{t}$, this notion is depicted in Fig. 1 The definition of two optimally aligned individuals is as follows.

Optimally Aligned Individuals: Two GP individuals A and B are optimally aligned if exist a scalar k such that $\overrightarrow{e}_A = k \cdot \overrightarrow{e}_B$.

The concept of optimally aligned individuals depicted in Fig. 2 allows for the calculation of the optimal solution analytically. Let A and B be two optimally aligned individuals. Then the equation of the aligned individuals can be rewritten as

$$\overrightarrow{S_A} - \overrightarrow{t} = k \cdot \left(\overrightarrow{S_B} - \overrightarrow{t}\right) . \tag{9}$$

Now, obtaining $\overrightarrow{t}$ is

$$\overrightarrow{t} = \frac{1}{1-k} \cdot \overrightarrow{S_A} - \frac{1}{1-k} \cdot \overrightarrow{S_B} . \tag{10}$$

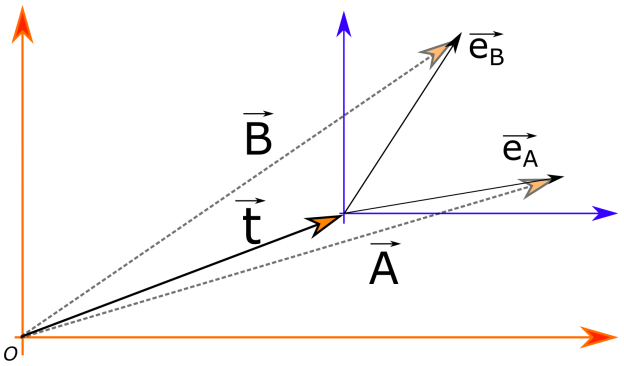

Fig. 1. A 2-D representation of the transformation from the semantic space to the error space. In practice semantic and error spaces are multidimensional with a number of dimensions being equal to the number of instances.

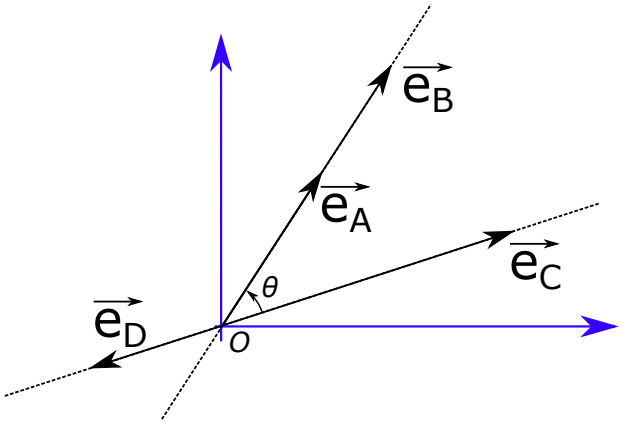

Fig. 2. A graphical 2-D representation of the error alignment: vector $\overrightarrow{e_A}$ is aligned with vector $\overrightarrow{e_B}$ and the vector $\overrightarrow{e_C}$ is aligned with vector $\overrightarrow{e_D}$. In practice semantic and error spaces are multi-dimensional with a number of dimensions being equal to the number of instances.

From where we can analytically construct the optimal solution if we consider that we are looking for the optimal solution k_{opt} with semantics $\overrightarrow{t}$, and if A and B represent the syntactic structure of the two aligned individuals. That is

$$k_{opt} = \frac{1}{1-k} \cdot A - \frac{1}{1-k} \cdot B. \tag{11}$$

This implies that if we are able to find two optimally aligned individuals A and B with semantics $\overrightarrow{S_A}$ and $\overrightarrow{S_B}$ it is possible to find the optimal solution syntax k_{opt} that corresponds to the semantics of $\overrightarrow{t}$ by the direct application of Eq. 11.

D. Reconstructing solutions using GSGP and Error Alignment

In this work, we stick to the definition of optimally aligned individuals in the error space made in [12]. GSGP and error alignment are used in conjunction to find two optimally aligned individuals and reconstruct the target directly, an approach that has been used in [30]. The goal in [12] is to minimize the angle between error vectors in order to find two aligned individuals, then, the optimal solution is found analytically; while in [30] the goal is to minimize the distance to a new target semantics by using a geometric semantic

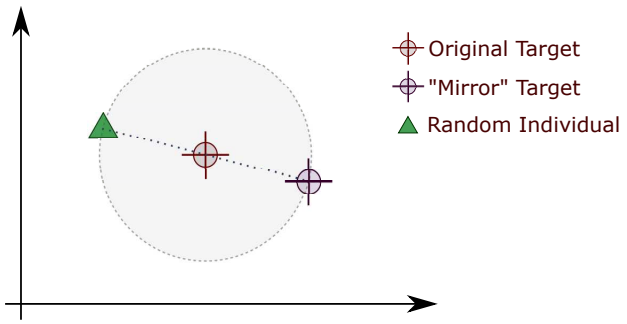

Fig. 3. Random individual and new target computation.

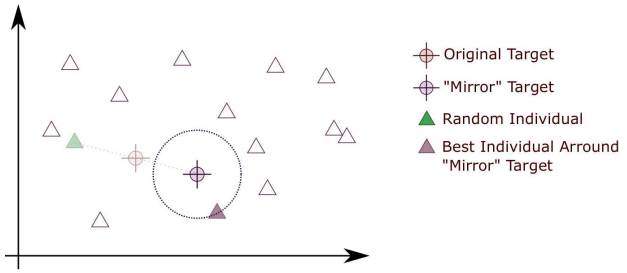

Fig. 4. Initial population for evolution around the new target.

hill climber to explore the search space. In our approach we find two optimally aligned individuals by searching for an individual that minimizes the distance to a "mirror target" which in turn is aligned to a randomly produced individual, this is done by applying the mutation with the OMS. Thus we split the problem in two parts. The first step is the generation of a random individual within a certain distance from the target, then calculate the semantics of the "mirror target" (see Fig. 3) using Eq. 9.

The second step will be to set the "mirror target" as the target of the problem, this is done in an attempt to minimize the risk of overfiting by searching for the "mirror target" instead of searching for the target directly. Afterwards, we run the GP process towards this mirror target (see Fig. 4). After that we solve the problem of finding the syntactic representation of a individual which minimizes the distance to the "mirror target" in semantic space. Then we use that solution to construct an optimal solution. The mutation utilizes the OMS in the process of searching for the "mirror target". The fitness used to guide the evolution is the distance relative to the "mirror target" in the semantic space, instead of using the original target as reference.

The third step will be to use the nearest individual to the "mirror target" to reconstruct a solution using the error alignment method (Eq. 11). The distance from the original target to the reconstructed individual is the error reported in the experimental section of this work since it is the performance on the actual problem target, Fig. 5 shows this step graphically. The process continues until a stop criterion is met. In this method, test fitness is computed based on the reconstructed solution after the error alignment procedure. Hereafter, this

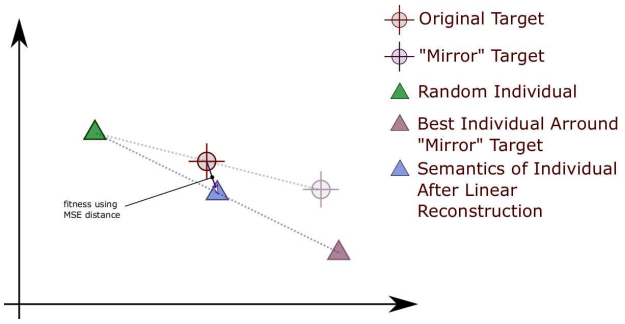

Fig. 5. Fitness evaluation with the best individual (after reproduction and mutation).

method will be referred to as GSGP-EA+OMS.

E. Hybrid Proposals

In this section we present our implementation of two hybrid methods based on the OMS, given its nice property of converging quickly in just few generations to low error values. We refer to these approaches as HyOMS-1 and HyOMS-2, which stand for **Hy**brid algorithm with **O**ptimal **M**utation **S**tep, with two versions.

In previous works, it has become apparent that the OMS method (and similarly GP-LS) converges quickly at the beginning, but then fails to improve any further. On the other hand, the standard GSM operator converges much slower at a steady pace across the entire run, and also does not seem to be affected by overfitting. Therefore, the proposed Hybrid approaches attempt to combine the use of both mutations.

The first attempt was to run the algorithm with the OMS at the begging for a few generations and then change to standard GSM. A similar approach was used in [13] but the problem is that it was not possible to determine how many generations should the search run with one mutation and then switch to the other one, thus this approach was discarded.

In HyOMS-1 we perform OMS during 10 generations and we store the fitness of each generations in a buffer, using a sliding window, this is because after 10 generations the effect of the OMS on the fitness tends to diminish considerably, but a wider window can be also used. Subsequently, we calculate the mean fitness improvement within the window. If the result is not lower than a certain threshold (i.e., the OMS is not performing any significant improvements) we start using GSM, otherwise the evolution continues to perform OMS until the performance improvement stagnates. This is a simple adaptation process to switch between the different mutation operators.

A different approach is taken in HyOMS-2. In this case both OMS and the standard GSM are always performed, and the mutation that provided the best improvement is kept while the other mutation is discarded. Hence, in HyOMS-2, the best fitness is taken by considering both methods. While this method incurs a slight increase in computation time, it is practically negligible given that these operations occur fully

Problem Set	Number of Training Cases	Number of Test Cases	Number of Features
Energy CL	538	230	8
Energy HL	538	230	8
Housing	354	152	12

TABLE I
NUMBER OF CASES AND FEATURES OF SELECTED PROBLEMS

Parameter	Value
Function Set	$+, -, /, *$
Population Size	200
Max. Number of Generations	2000
Probability of Crossover	0.0 %
Probability of Mutation	100 %
Max. Depth of Initial Trees	6
Tournament Size	4
Mutation Step	OMS, GSM (0.1)
Fitness Measurement	MAE

TABLE II
CONFIGURATION OF THE GSGP RUNS.

in semantic space and none of the mutations are complex or time consuming.

IV. EXPERIMENTAL RESULTS

This section presents our experimental work, detailing the test problems, the tested algorithms, our experimental setup and implementation details. Afterward, we present a detailed discussion of the main results obtained.

A. Datasets and experimental setup

In total, the experimental evaluation includes three common real-world problems used in machine learning and symbolic regression literature.

We used two datasets used to predict the energy consumption of residential buildings based on the heating load (Energy HL) and cooling load (Energy CL) [18]. These datasets take into account the insulation and construction of the building, including floors, walls, ceilings and roofs, and the building's glazing and skylight based on size, performance, shading and overshadowing. The correct estimation of HL and CL are very important to achieve an efficient heating, ventilation and air-conditioning (HVAC) in building designs. These datasets have been used with GSGP before, with strong results relative to other learning methods [18].

The third datset is referred to as Housing, which is used to predict the housing value in the city of Boston, MA in the USA [24]. This dataset takes into account variables like per capita crime rate by town, index of accessibility to radial highways, pupil-teacher ratio by town, among other important measures. Table I summarizes the number of fitness cases and features for each of the tested problems.

The common parameters of the GP algorithms are shown in Table II, this is the configuration we have used for all runs.

In total, the following algorithms are tested: (1) GSGP; (2) GSGP with Optimal Mutation Step (GSGP + OMS); two different Hybrid methods with the OMS, namely (3) HyOMS-1 and (4) HyOMS-2; and (5) GSGP with Error Alignment and OMS, named GSGP-EA+OMS. All algorithms are based on the GSGP implementation in [16], which uses tables of references (pointers) in C++ with a fixed step mutation of 0.1 and without crossover. The switching threshold between OMS to GMS used in HyOMS-1 was set to 0.001.

For each problem, we present two plots to analyze the results. First, convergence plots of the median training fitness over 30 independent runs using random partitions of the data, with a 70-30 split between training and testing. The second plot is a boxplot comparison of the training and testing performance, to assess the amount of overfitting by each method as well as the differences between the performance of all the methods. These results are summarized in Figures 6 to 11. Moreover, a numerical summary of the results are provided in Tables III and IV that present the median fitness at the last generation (2000th generation) obtained for the three problems. Table III presents the training performance, while Table IV presents the testing performance.

B. Results and discussion

In terms of convergence, shown in Figures 6, 8 and 10, it is evident that most of the methods that use the OMS converge much faster than GSGP.

Some other trends include the following. First, GSGP converges slower, but reaches the same performance (or slightly better) on two problems (Energy CL and Energy HL) than GSGP + OMS and GSGP-EA+OMS. Second, both GSGP+OMS and GSGP-EA+OMS converge quickly and seem to stagnate after a small number of generations, except for the Housing problem where slight improvements are still produced across all generations. On the other hand, both hybrid approaches show a clear trend towards incremental improvements throughout the search process, in some cases the difference with the other methods seems substantial (Energy CL and Energy HL). These results are confirmed in Table III, where the best median performance is exhibited by HyOMS-2, followed by HyOMS-1. This behaviour should be expected, since these hybrid approaches are using a greedy approach to discard offspring or to switch the mutation operator during the run.

In the boxplot comparisons of Figures 7, 9 and 11, we can see the following. First, all methods exhibit a slight effect of overfitting, with training performance always being better (sometimes only marginally so) than testing performance. The uniformity is encouraging, however, because even the most greedy hybrid approach (HyOMS-2) does not seem to deviate from the other methods. Second, on almost all problems the largest variance and worst performance is exhibited by GSGP-EA+OMS, in many cases producing outlier results that are not shown given the scale of the plots. Of all the other methods, GSGP performs the worst, while HyOMS-2 consistently shows the best median training and testing performance, also summarized in Table IV. Finally, the hybrid methods exhibit clear performance improvements relative to the other methods on

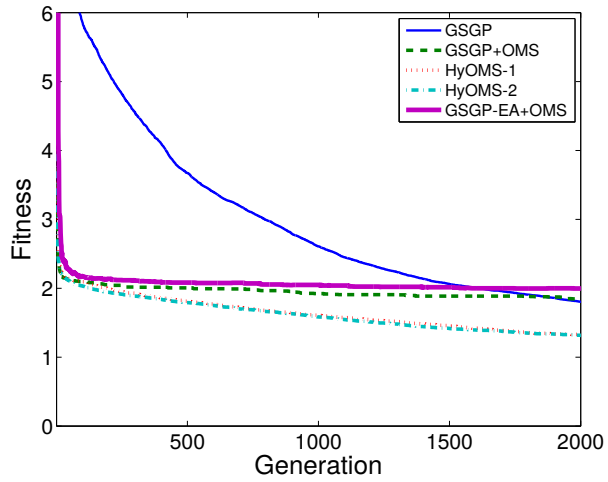

Fig. 6. EnergyCL: median of test fitness at each generation.

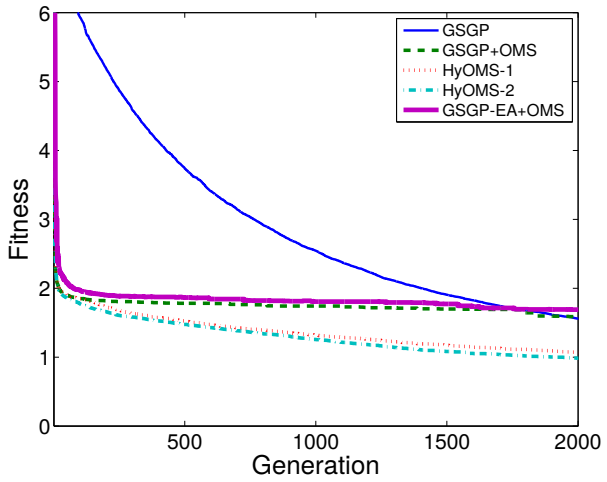

Fig. 8. EnergyHL: median of test fitness at each generation.

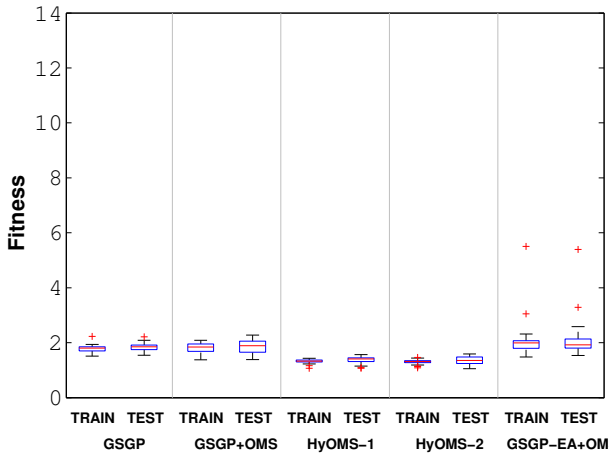

Fig. 7. EnergyCL: Box-plot comparison (not all outliers are shown).

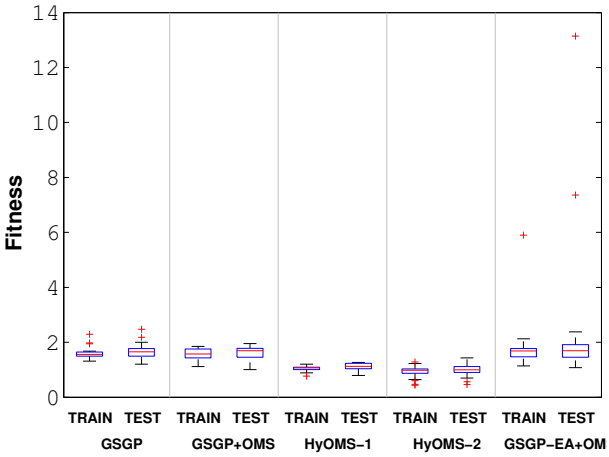

Fig. 9. EnergyHL: Box-plot comparison (not all outliers are shown).

the Energy HL and Energy CL problems, which is particularly encouraging since it is also the problems on which overfitting is practically non-existent for these methods.

Finally, it is instructive to analyze the behavior of HyOMS-2 since it achieved the best performance overall. We are interested in quantifying the number of times that either standard GSM was applied or by using the OMS; i.e., the percentage of mutations where the best child was produced by standard GSM or OMS. Figures 12, 13 and 14 show these results, as the median value of all runs for each problem. Note that in figure 12, for problem Energy CL, and figure 14 for problem Housing, the tendency is to mainly perform OMS at the beginning of the run and then switch to GSM. This is consistent with the proposal in [13], where GSM with a local search is applied for the first generations and than standard GSM afterwards. However, OMS is still successful in some cases (roughly $< 8\%$) after the initial generations,

TRAINING MAE			
	energyCL	energyHL	housing
GSGP	1.8036	1.5575	3.3542
GSGP+OMS	1.8444	1.5761	2.36
HyOMS-1	1.3266	1.0688	2.4209
HyOMS-2	**1.3147**	**0.9832**	**2.0435**
GSGP-EA+OMS	1.9961	1.6876	2.633

TABLE III
MEDIAN TRAINING FITNESS FOR EACH METHOD ON EACH PROBLEM, WITH BOLD INDICATING THE BEST RESULT. MEDIAN OVER 30 RUNS.

TRAINING MAE			
	energyCL	energyHL	housing
GSGP	1.8524	1.6563	3.941
GSGP+OMS	1.8926	1.6925	3.1437
HyOMS-1	1.4077	1.1225	3.2366
HyOMS-2	**1.3516**	**0.9975**	**3.064**
GSGP-EA+OMS	1.9188	1.695	3.5515

TABLE IV
MEDIAN OF TEST FOR THE FIVE METHODS, WITH BOLD INDICATING THE BEST RESULT. MEDIAN OVER 30 RUNS.

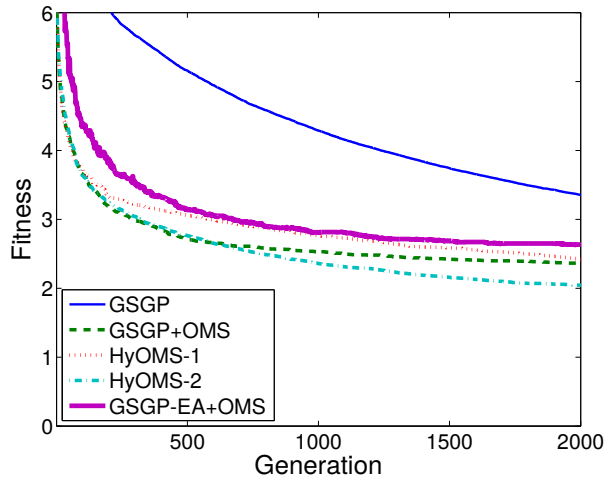

Fig. 10. Housing: median of test fitness at each generation.

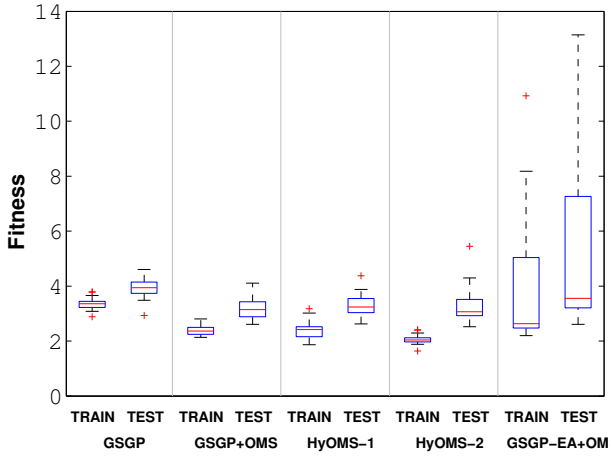

Fig. 11. Housing: Box-plot comparison (not all outliers are shown)

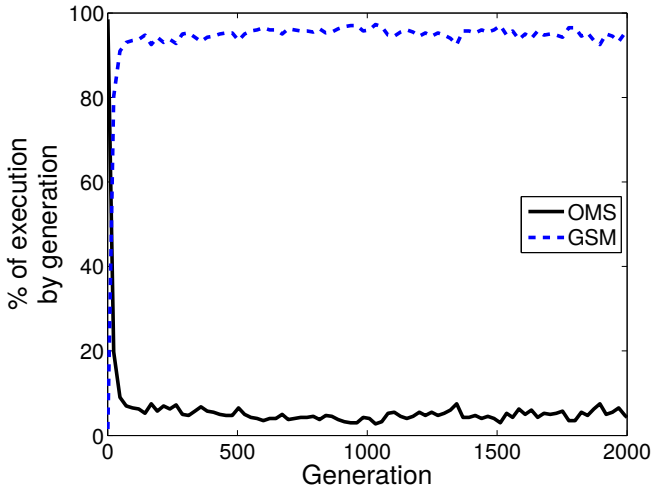

Fig. 12. Energy CL: percentage of times that either standard GSM or OMS generated the best offspring. The plots show the median values over 30 runs.

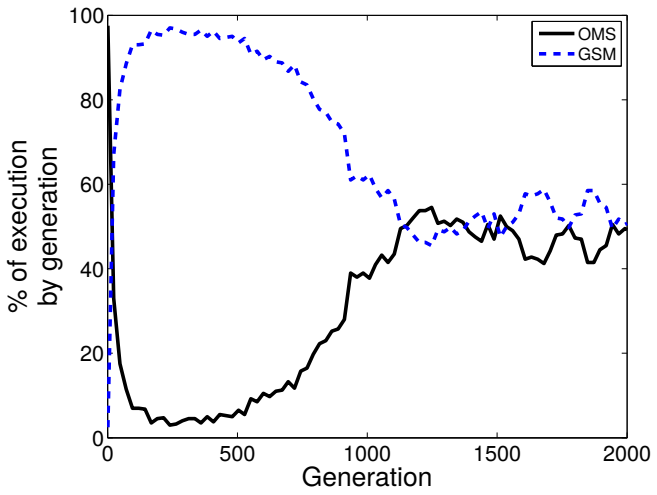

Fig. 13. Energy HL: percentage of times that either standard GSM or OMS generated the best offspring. The plots show the median values over 30 runs.

so the use of both mutation operators seems justified. On the other hand, for the Energy HL problem, shown in figure 13, the behavior of the algorithm is quite different. At the beginning, when the solutions are far from the target then OMS allows for a quicker convergence and is thus used more frequently. Afterwards, the OMS estimation stagnates and standard GSM (which is more explorative) is preferred, again consistent with the results on the other two problems. The difference appears at about generation 1000 (half of the run), where both methods are applied with almost the same proportion. Once again, this strongly indicates that using both methods concurrently is the main reason why HyOMS-2 achieves the best performance.

V. Conclusions and Future Work

This paper is part of a broader research project, aimed at developing a hardware implementation of a full GP system using FPGAs. Given the particularities of the GSGP approach, we believe that it presents a perfect candidate for this stated goal. As a first step, in this work we have evaluated several recent GSGP variants, that integrate greedy local search methods or heuristics to improve algorithm convergence and performance.

We have compared five variations of the GSGP algorithm, namely: (1) GSGP; (2) GSGP with Optimal Mutation Step (GSGP + OMS); two different Hybrid methods with the OMS, namely (3) HyOMS-1 and (4) HyOMS-2; and (5) the GSGP with Error Alignment and OMS, named GSGP-EA+OMS. Results show that both hybrid methods achieve the best performance, and might present suitable approaches for the FPGA implementation. HyOMS-2 exhibited the best performance, but HyOMS-1 performance was very similar and is a bit more efficient since it only utilizes a single mutation operator, but this difference might be negligible in a parallel implementation.

Future work, will focus on implementing the full GSGP on an FPGA device, and in particular to implement the proposed HyOMS-1 and HyOMS-2 variants. However, it must

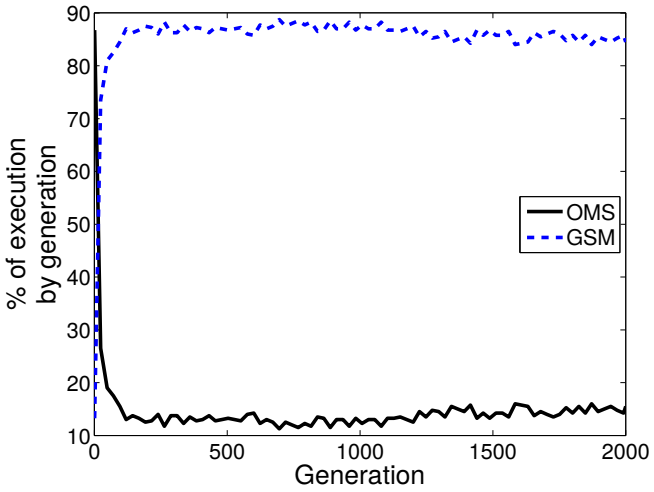

Fig. 14. Housing: percentage of times that either standard GSM or OMS generated the best offspring. The plots show the median values over 30 runs.

be stressed that this will not be a trivial endeavour, requiring the solution of complex tasks for an FPGA device, including program representation, genetic operators, fitness evaluation and other tasks that are trivial when done in software but can be quite complex operating directly on hardware.

ACKNOWLEDGMENT

First authors was supported by CONACYT (Mexico) scholarship 406989. Funding was provided by CONACYT (Mexico) project FC-2015-2/944 "Aprendizaje evolutivo a gran escala". Authors want to thank the Tecnologico Nacional de Mexico for its support with the project No. 5861.16-F.

REFERENCES

[1] R. Poli, W. B. Langdon, and N. F. McPhee. A Field Guide to Genetic Programming. Lulu Enterprises, UK Ltd. 2008.

[2] R. Poli, N F. McPhee, L. Vanneschi. Analysis of the Effects of Elitism on Bloat in Linear and Tree-based Genetic Programming. *Genetic Programming Theory and Practice VI, Chap. 7, pp. 91-111*, 2008.

[3] D. A. Augusto, H. J.C. Barbosa. Accelerated parallel genetic programming tree evaluation with OpenCL. *Journal of Parallel and Distributed Computing, Vol. 73, Issue 1, pp. 86-100*, January 2013.

[4] S. Harding. Evolution of image filters on graphics processor units using cartesian genetic programming. *Evolutionary Computation, IEEE World Congress on Computational Intelligence, pp. 1921-1928*, 2008.

[5] S. Harding, W. Banzhaf. Fast genetic programming on GPUs. *Proceedings of the 10th European Conference on Genetic Programming (EuroGP'07), pp. 90-101*, 2007.

[6] D. Perry. VHDL : Programming By Example. McGraw-Hill Education, 2002.

[7] J. Deschamps, G. Sutter, E. Cant. Guide to FPGA Implementation of Arithmetic Functions. Springer , 2012.

[8] B. Scheuermann, K. So, M. Guntsch, M. Middendorf, O. Diessel, H. ElGindy, H. Schmeck. FPGA implementation of population-based ant colony optimization. *Applied Soft Computing, Volume 4, No. 3, pp. 9303 - 322*, 2004.

[9] Spartan-6 FPGA Configurable Logic Block. *Xilinx*, 2010.

[10] Y. Jin A Comprehensive Survey of Fitness Approximation in Evolutionary Computation. *Soft Computing, Vol.9, No. 1, pp. 3-12*, 2005.

[11] A. Moraglio, K. Krawiec and C. G. Johnson. Geometric semantic genetic programming. *Proceedings of the 12th International Conference on Parallel Problem Solving from Nature, Vol. I, pp.21-31*, 2012.

[12] S. Ruberto, L. Vanneschi, M. Castelli and S. Silva. ESAGP – A Semantic GP Framework Based on Alignment in the Error Space. *EuroGP 2014, LNCS 8599, pp. 150-161*, 2014.

[13] M. Castelli, L. Trujillo, L. Vanneschi, S. Silva, E. Z-Flores and P. Legrand. Geometric Semantic Genetic Programming with Local Search. *GECCO '15, pp. 999-1006*, 2015.

[14] M. Castelli, I. Gonçalves, L. Trujillo, A. Popovič. An evolutionary system for ozone concentration forecasting. *Information Systems Frontiers, pp. 1-10*, 2016.

[15] M. Castelli, L. Trujillo, L. Vanneschi, A. Popovič. Prediction of relative position of CT slices using a computational intelligence system. *Applied Soft Computing, Vol. 46, pp. 537-542*, 2016.

[16] M. Castelli, S. Silva and L. Vanneschi. A C++ framework for geometric semantic genetic programming. *Genetic Programming and Evolvable Machines, Vol. 16, Issue 1, pp. 73-81*, 2015.

[17] I. Gonçalves, S. Silva and C.M. Fonseca On the Generalization Ability of Geometric Semantic Genetic Programming. *Genetic Programming: 18th European Conference, EuroGP 2015, Proceedings, pp. 41-52*, 2015.

[18] M. Castelli, L. Trujillo, L. Vanneschi, A. Popovič. Prediction of energy performance of residential buildings: A genetic programming approach. *Energy and Buildings, Vol. 102, pp. 67-74*, 2015.

[19] M. S. Brown, and M. J. Pelosi, and H. Dirska. Dynamic-radius Species-conserving Genetic Algorithm for the Financial Forecasting of Dow Jones Index Stocks. *Proceedings of the 9th International Conference on Machine Learning and Data Mining in Pattern Recognition, pp. 27-41*, 2013.

[20] M. Brameier and W. Banzhaf. A Comparison of Linear Genetic Programming and Neural Networks in Medical Data Mining. *IEEE Transactions on Evolutionary Computation, Vol. 5, pp. 17-26*, 2001.

[21] M. Brameier and W. Banzhaf. Explicit Control of Diversity and Effective Variation Distance in Linear Genetic Programming. *Genetic Programming: 5th European Conference, EuroGP 2002, pp. 37-49*, 2002.

[22] J. F. Miller, P. Thomson. Cartesian Genetic Programming. *Genetic Programming: European Conference, EuroGP 2000, pp. 121-132*, 2000.

[23] T. Perkis. Stack Based Genetic Programming. *Proceedings of the 1994 IEEE World Congress on Computational Intelligence 1994, Vol. 1, pp. 148-153*, 1994.

[24] J. R. Quinlan Combining Instance-Based and Model-Based Learning. *Machine Learning, Vol. 76, pp. 236-243*, 1993.

[25] L. Spector. Autoconstructive Evolution: Push, PushGP, and Pushpop. *Proceedings of the Genetic and Evolutionary Computation, GECCO-2001, pp. 137146*, 2001.

[26] C. Goribar, Y. Maldonado, L. Trujillo. Random Tree Generator for an FPGA-based Genetic Programming System. *Proceedings of the Genetic and Evolutionary Computation, GECCO-2016, pp. 1023-1026*, 2016.

[27] C. Goribar, Y. Maldonado, L. Trujillo. Automatic Random Tree Generator on FPGA. *NEO 2015: Results of the Numerical and Evolutionary Optimization Workshop NEO 2015, pp. 89-104*, 2017.

[28] L. Vanneschi, M. Castelli, L. Manzoni and S. Silva. A New Implementation of Geometric Semantic GP and Its Application to Problems in Pharmacokinetics. *Proceedings of the 16th European Conference on Genetic Programming, EuroGP'13, pp. 205-21*, 2013.

[29] R. Prez-Andrade, C. Torres-Huitzil, R. Cumplido Processor arrays generation for matrix algorithms used in embedded platforms implemented on FPGAs. *Microprocessors and Microsystems, Vol. 39, pp. 576-588*, 2015.

[30] I. Gonçalves, S. Silva, C. Fonseca, M. Castelli, Arbitrarily Close Alignments in the Error Space: A Geometric Semantic Genetic Programming Approach *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference Companion, pp. 99-100*, 2016.