A Comparison between Geometric Semantic GP and Cartesian GP for Boolean Functions Learning*

Andrea Mambrini
School of Computer Science
University of Birmingham
B15 2TT, Birmingham, United Kingdom
a.mambrini@cs.bham.ac.uk

Luca Manzoni
Univ. Nice Sophia Antipolis, CNRS, I3S
UMR 7271, 06900 Sophia Antipolis, France
luca.manzoni@i3s.unice.fr

ABSTRACT

Geometric Semantic Genetic Programming (GSGP) is a recently defined form of Genetic Programming (GP) that has shown promising results on single output Boolean problems when compared with standard tree-based GP. In this paper we compare GSGP with Cartesian GP (CGP) on comprehensive set of Boolean benchmarks, consisting of both single and multiple outputs Boolean problems. The results obtained show that GSGP outperforms also CGP, confirming the efficacy of GSGP in solving Boolean problems.

Keywords

Geometric Semantic Genetic Programming, Cartesian Genetic Programing, Boolean functions

1. INTRODUCTION AND BASIC NOTIONS

Genetic Programming (GP) was introduced as a method to evolve computer programs in a way similar to classical Genetic Algorithms. In this work we compare *Geometric Semantic GP* (GSGP), a recently defined kind of GP based on semantic operators introduced by Moraglio and coworkers [5], with Cartesian GP (CGP) (see [3] for a comprehensive overview). Since its definition, GSGP has been compared to traditional tree-based GP [5] with the standard subtree crossover and mutation operators, on single-output Boolean problems. In order to assess the performances of GSGP in a more exhaustive way, this work compares GSGP and GSGP with a particular variant of the mutation operator, called block mutation [1], with CGP on a comprehensive set of Boolean benchmarks. The results will show that for both kinds of benchmarks, GSGP and GSGP with block mutation outperform CGP, confirming the efficacy of GSGP in solving Boolean problems even when compared with non-tree-based GP and on multiple-output benchmarks.

*This work has been partially supported by the French National Research Agency project EMC (ANR-09-BLAN-0164) and by EPSRC (Grant No. EP/I010297/1).

GECCO'14, July 12–16, 2014, Vancouver, BC, Canada.
ACM 978-1-4503-1964-5/14/07
http://dx.doi.org/10.1145/2598394.2598475 .

2. EXPERIMENTAL SETTINGS

To compare CGP, GSGP, and GSGP with block mutation we have used the following 4 benchmark problems: k-even parity (with k ranging from 4 to 6), k-multiplexer ($k = 3$ and $k = 6$), digital adder (2 and 3 bits), and digital multiplier (2 and 3 bits). The inclusion the last two problems, which are multiple-output, follows the suggestion of [2] and [7].

For CGP we have used as a base the implementation by Julian Miller[1]. The parameter settings were derived from [4], in which an high number of nodes and a low mutation probability were found to be effective. Hence, we used 4000 nodes with a mutation rate of 1%. The population size was 5 and each individual could use AND, OR, NOR, and NAND as actions for the functional units.

For GSGP we have used a $(1 + 1)$EA setting [6]: the population is composed of one individual, at every generation the unique individual in the population is mutated and, if the resulting tree has a better fitness than its parent, it replaces the current individual. Since the population has size 1, we don't perform crossover. We compared Forcing Point Mutation (see Definition 3 in [1]) with Multiple Size Block Mutation (see Definition 9 in [1]). The starting individual is an empty tree, returning "false" on all inputs. To produce multiple outputs with GSGP we have represented every individuals as an array of standard (i.e., single-output) GSGP trees, each one providing one output. The mutation is performed by randomly selecting a position of the array and then mutating the tree in that position.

We have performed 100 independent runs for every benchmark problem using a training set comprising all possible inputs (i.e., a complete training set). We have used as a fitness the fraction of correct output bits. Notice that, for multiple-outputs problem, this fitness does not always correspond to the fraction of correct outputs (since one output consist in more than a bit). The evolution was stopped after reaching a perfect score, i.e., all correct outputs, on the training set.

[1] `http://www.cartesiangp.co.uk/resources/CGP-version1_1.7z`

		4-even parity	5-even parity	6-even parity	3-multiplexer	6-multiplexer	2-bits adder	3-bits adder	2-bits multiplier	3-bits multiplier
GSGP	*Average*	85.8	217.9	484.4	33.5	502.2	849.3	6200.5	387.7	4099.2
	Std. Dev.	38.0	88.7	128.5	21.2	154.8	194.6	1100.4	144.8	1049.3
	Median	76.0	198.5	473.0	28.5	453.5	825.0	6096.0	365.0	3874.5
	90^{th} *percentile*	136.0	323.0	628.5	57.5	765.0	1107.5	7521.0	583.0	5313.5
GSGP (block)	*Average*	372.1	1102.1	2967.6	49.0	706.6	2209.8	19580.5	949.9	12116.9
	Std. Dev.	163.6	419.1	860.4	30.4	284.7	743.0	5259.7	471.5	4445.3
	Median	360.0	977.0	2916.0	42.0	670.0	2157.5	18590.0	810.5	11312.5
	90^{th} *percentile*	585.5	1715.0	4027.0	93.5	1117.0	3146.5	26964.5	1552.5	18597.5
CGP	*Average*	2450.7	8447.3	24722.5	161.8	1487.5	13147.2	110693.0	1898.3	296260.0
	Std. Dev.	2848.8	5764.8	17486.0	314.0	2304.6	10012.8	80568.4	1754.4	169132.0
	Median	1727.0	7164.5	21886.0	88.0	1041.0	10628.5	86477.0	1493.5	240308.0
	90^{th} *percentile*	4970.0	16658.5	39388.5	322.5	2509.0	26457.5	219518.0	3708.0	521820.5

Table 1: A summary of the results. For each problem and for each method, the average, standard deviation, median, and the 90^{th} percentile of the number of generations needed to reach the optimum are reported.

3. EXPERIMENTAL RESULTS

We report the results of the experiments in Table 1. We have performed a Mann-Whitney U-test (see [8] for a description) with a significance level of 0.01 under the alternative hypothesis that the GSGP (resp., GSGP with block mutation) find the optimum before CGP with probability greater than one half. In all cases GSGP and GSGP with block mutation performed significantly better than CGP.

In all problems tested we have observed the same pattern of behavior from the three tested methods. GSGP was the best performer in all problems, GSGP with block mutation the second best, and CGP was in all cases the worst performer. From the high standard deviation and the gap between median and average we can observe the strong presence of outliers in CGP. Such problem is absent in both GSGP and GSGP with block mutation, for which the number of generations is closer to the average. It is interesting to note that, in all benchmarks, as the problem difficulty increases the gap between GSGP (resp., GSGP with block mutation) gets wider, i.e., the two semantic methods seems to scale better.

4. CONCLUSIONS

In this paper we have performed a comparison between Cartesian GP and Geometric Semantic GP with two different mutation operators on a wide range of single and multiple outputs problems. In all the considered problems GSGP is able to find the optimum using less generations than the other methods. GSGP with block mutation ranks second, while CGP has the worst performances in all the benchmark problems. Moreover the standard deviation of the number of generations to find the optimum for both the GSGP operators tested is lower than the one for CGP, making the performance of GSGP more stable. The performance gap between the three methods is statistically significant for all the considered benchmark problems. This result confirms GSGP as a good method to solve Boolean problems.

Possible future works can compare GSGP with more advanced variants of CGP and investigate the use of semantic crossover in GSGP. Another important line of research to assess the performance of GSGP as a learning tool, is to investigate how the formula evolved from a small training set generalize on unseen input.

5. REFERENCES

[1] A. Mambrini, L. Manzoni, and A. Moraglio. Runtime analysis of mutation-based geometric semantic genetic programming on boolean functions. In *Foundations of Genetic Algorithm - FOGA 2013*, pages 119–132, Adelaide, Australia, January 2013. ACM.

[2] J. McDermott, D. R. White, S. Luke, L. Manzoni, M. Castelli, L. Vanneschi, W. Jaskowski, K. Krawiec, R. Harper, K. De Jong, and U.-M. O'Reilly. Genetic programming needs better benchmarks. In *Proceedings of the fourteenth international conference on Genetic and evolutionary computation conference*, GECCO '12, pages 791–798, New York, NY, USA, 2012. ACM.

[3] J. F. Miller. *Cartesian genetic programming*. Springer, 2011.

[4] J. F. Miller and S. L. Smith. Redundancy and computational efficiency in cartesian genetic programming. *IEEE Transactions on Evolutionary Computation*, 10(2):167–174, April 2006.

[5] A. Moraglio, K. Krawiec, and C. Johnson. Geometric semantic genetic programming. In *PPSN '12*, volume 7491 of *Lecture Notes in Computer Science*, pages 21–31. Springer, 2012.

[6] F. Neumann and C. Witt. *Bioinspired Computation in Combinatorial Optimization: Algorithms and Their Computational Complexity*. Springer, 2010.

[7] D. White, J. McDermott, M. Castelli, L. Manzoni, B. Goldman, G. Kronberger, W. Jaśkowski, U. O'Reilly, and S. Luke. Better gp benchmarks: community survey results and proposals. *Genetic Programming and Evolvable Machines*, 14(1):3–29, 2013.

[8] D. A. Wolfe and M. Hollander. *Nonparametric statistical methods*. John Wiley New York, 1973.