Preliminary Study of Bloat in Genetic Programming with Behavior-Based Search

Leonardo Trujillo, Enrique Naredo and Yuliana Martínez

Doctorado en Ciencias de la Ingeniería, Departamento de Ingeniería Eléctrica y Electrónica, Instituto Tecnológico de Tijuana, Blvd. Industrial y Av. ITR Tijuana S/N, Mesa Otay C.P. 22500, Tijuana B.C., México,
leonardo.trujillo@tectijuana.edu.mx, enriquenaredo@gmail.com, ysaraimr@gmail.com

Abstract. Bloat is one of the most interesting theoretical problems in genetic programming (GP), and one of the most important pragmatic limitations in the development of real-world GP solutions. Over the years, many theories regarding the causes of bloat have been proposed and a variety of bloat control methods have been developed. It seems that one of the underlying causes of bloat is the search for fitness; as the fitness-causes-bloat theory states, selective bias towards fitness seems to unavoidably lead the search towards programs with a large size. Intuitively, however, abandoning fitness does not appear to be an option. This paper, studies a GP system that does not require an explicit fitness function, instead it relies on behavior-based search, where programs are described by the behavior they exhibit and selective pressure is biased towards unique behaviors using the novelty search algorithm. Initial results are encouraging, the average program size of the evolving population does not increase with novelty search; i.e., bloat is avoided by focusing on novelty instead of quality.

Keywords: Bloat, Genetic Programming, Novelty Search

1 Introduction

Genetic programming (GP) has shown to be an effective search method for automatic program induction, with noteworthy results in many domains [10]. Nonetheless, most GP practitioners will most likely have to overcome several issues in order to apply the paradigm successfully. For instance, GP has many degrees of freedom that require a proper initialization and parametrization, an issue for all evolutionary algorithms (EAs). However, probably the most studied GP problem, is the way in which solutions grow in size as the search progresses, what is known as bloat. Stated more precisely, bloat is excessive code growth within the individuals of the evolving population without a proportional improvement in fitness. Over the years, many bloat theories have been developed and many bloat control methods have been proposed [31, 32]. One of the most promising attempts at explaining the bloat phenomenon is the crossover bias theory [5, 28], that has lead to a powerful bloat control method called operator equalisation [32]. However, recent experimental results have blurred what is understood regarding the causes of bloat [29], as well as what are the best strategies that can be used to eliminate it from GP runs [7, 29].

This work revisits the fitness-causes-bloat theory of Langdon and Poli [13, 14, 31], that basically states that the search for better fitness will bias the search towards larger trees, simply because there are more large programs than small ones. Silva and Costa [31] state it clearly:

> ... one cannot help but notice the one thing that all the [bloat] theories have in common, the one thing that if removed would cause bloat to disappear, ironically the one thing that cannot be removed without rendering the whole process useless: the search for fitness.

This paper presents a preliminary study that supports the fitness-causes-bloat theory in GP. It is shown that, for the test problems presented here (supervised data classification), bloat can be avoided by abandoning an explicit fitness function. In other words, by not searching for fitness directly the bloating effect is eliminated. Moreover, the proposed approach is not *useless*, in fact it is quite competitive with standard fitness-based search. The proposal is to use a behavior-based search with GP applying the novelty search (NS) algorithm, substituting an explicit fitness function by an optimization criterion that is biased towards solutions that are unique, or novel, with respect to the rest of the population. Indeed, experimental results confirm that by eliminating an explicit fitness bias from the search then GP is not bloated, and the quality of the solutions is not compromised, particularly for hard problems.

The remainder of the paper proceeds as follows. Section 2 provides a brief overview on bloat, discussing recent theories and bloat control methods. The concept of behavior-based search and behavioral space is explored in Section 3, along with the NS algorithm. Section 4 describes a NS-based GP for data classification as a case study for bloat analysis. Afterwards, experimental results are presented and discussed in Section 5. Finally, a summary, conclusions and future work are given in Section 6.

2 Recent Advances in Bloat

Over the last twenty years, many theories have been put forth to explain bloat, attempting to understand why it happens, and to propose strategies to eliminate it. Many bloat control methods have been developed, focusing on modifying different aspects of the evolutionary process, such as the genetic operators, selection and survival strategies and fitness assignment. For a comprehensive review on previous theories and methods, the reader is referred to the work by Silva and Costa [31].

Currently, the most plausible theory for bloat is the crossover bias theory (CBT), proposed by Dignum and Poli [5, 28]. Focusing on standard Koza style GP with a tree representation [11], the CBT states that bloat is produced by the effect that subtree crossover has on the distribution of tree sizes in the population. While the average tree size is not affected by crossover, the distribution of tree sizes is modified. Subtree crossover produces a large number of small trees, and since small trees represent trivial individuals in most problems, their fitness values are usually very bad. Therefore, selection will favor larger trees, causing an increase in the average size of trees within the population, effectively bootstrapping the bloating effect. This theory seems very promising, and experimental data supports it [32]. Moreover, powerful bloat control methods have been developed based on CBT, namely operator equalisation [32]. However, recent studies have made the

matter significantly less clear. For instance, Silva [29] suggests that some of the properties of operator equalisation that allow it to provide a bloat-free GP search, might not be consistent with the CBT. Harper [7] also has shown that operator equalisation is not the best bloat control method on some problems, and proposes other, more elaborate, strategies to eliminate bloat from GP runs. Finally, it is important to point out that even though operator equalisation can limit bloat for some problems, it is a computationally expensive algorithm, so developing other methods remains a worthwhile endeavor.

This work revisits another bloat theory, which can be called the fitness-causes-bloat theory (FCBT), developed by Langdon and Poli [13, 14]. The main arguments of FCBT proceed as follows [13, 14, 32]. For a variable length GP representation (such as a tree based GP) many genotypically different programs, of different sizes, can produce the same outputs on a given set of fitness cases. Therefore, all of these programs trees will be assigned the same fitness value. Then, because GP crossover tends to be a predominantly destructive search operator, when improved solutions are difficult to find then selection is biased towards offspring that have the same fitness as their parents. Since there are exponentially more large programs than small ones, an almost unavoidable tendency towards larger, or bloated, programs is present during a GP search.

This work attempts to shed some light on this matter. If bloat is a natural consequence of the search for better fitness, as Langdon and Poli, and Silva and Costa stated; then a natural bloat control strategy would be one where fitness is abandoned. However, the question then becomes: if fitness is not used to determine selection pressure, then what could be used in its stead? This question is not as strange as at first it may appear. Consider that EAs are inspired on biological Darwinian evolution, a natural process where an a priori purpose, or objective, is not present; i.e., there is not an explicit objective function in natural evolution. Natural evolution is an open-ended search process, where the search for fitness is not explicitly carried out, it is a natural consequence of the evolutionary dynamics induced by physical laws and chemical reactions. Inspired by nature, some researchers have also proposed open-ended EAs, dating back to the origins of the field [4], and other more recent examples [6, 9, 27]. In particular, this work studies the effect of bloat on one of the most recently developed open-ended EAs, the novelty search algorithm, where fitness is substituted by a measure of solution novelty. It is hypothesized that, if the FCBT is correct, then a NS-based GP would have to produce a significant reduction in bloat, or possibly a complete elimination of it.

3 Behavior-based Search

The main goal of any EA, is to search for the solution that achieves the best possible performance; hence, a proper measure of performance needs to be proposed for each problem. In the GP case, performance is normally computed based on a set of fitness cases or training set of data. Traditional fitness-based approaches will provide a single measure that characterizes the performance of an individual; for instance, the mean error with respect to a desired output. This is a coarse view of a program's performance, usually averaging out performance variations that a program might have on different fitness cases. However, it is not the only possible alternative, below two other approaches are discussed: semantics and behaviors.

Semantics in GP describes the performance of a program with the raw output vector computed over all fitness cases [21, 1, 2, 12, 20, 36]. Given a set of n fitness cases, the semantics of a program K is the corresponding output vector it produces $\mathbf{y} \in \mathbb{R}^n$. In GP, many genetically (phenotypically) different programs can share the same semantic output. Therefore, semantics adds another space of analysis in which the search is being conducted, along with genotypic, phenotypic and objective (fitness) space, we can also consider semantic space; where a many-to-one relation will usually exists between genotypic (phenotypic) space and semantic space.

Researchers have used semantics to improve GP in different ways, such as modifying traditional genetic operators to improve the semantic diversity of the evolving population [1, 2, 36], or by explicitly performing evolution within semantic space [12, 21]. In general, all of these works have shown improved results using a canonical GP as a control method, mostly on symbolic regression problems.

However, strictly focusing on program outputs might not be the best approach in some domains. For example, consider the GP classifier based on static range selection (SRS) [38] (it will be further discussed in Section 4 and used in the experimental work), that functions as follows. For a two class problem and real-valued GP outputs, the SRS classifier is straightforward; if the program output for input pattern $\mathbf{x}$ is greater than zero then the pattern is labeled as belonging to class A, otherwise it is labeled as a class B pattern. In this case, while the semantic space description (as defined above) of two programs might be different (maybe substantially), they can still produce the same classification for the input pattern.

Now, consider the case of evolutionary robotics (ER). In ER, evolution is normally used to search for neuro-controllers for autonomous robots [26]. The goal is to find robust solutions with good performance, while introducing as little prior knowledge as possible into the fitness function, such that the search is performed based on a very high-level definition of the task which needs to be solved [25]. Therefore, in ER the correspondence between program inputs, outputs and induced actions is not straightforward. Moreover, in ER fitness evaluation can be performed within real or simulated environments, where noisy sensors and the physical coupling between actuators and the real world, can produce a non-injective or non-surjective relation between program output and robot actions.

Therefore, some researchers have turned towards explicitly considering behavioral space [22, 34]. In robotics, the concept of behaviors dates back to the seminal works of R. Brooks in behavior-based robotics [3]. A behavior is a description β of the way an agent K (program in the GP case) acts in response to a series of stimuli within a particular context $\mathscr{C}$. A context $\mathscr{C}$ includes the description that an agent has about its own internal state and the characteristics of the surrounding environment at a given moment in time. Stated another way, a behavior β is produced by the interactions of agent K, output $\mathbf{y}$ and context $\mathscr{C}$. In behavior-based robotics, behaviors are described at a very high level of abstraction by the system designer. Conversely, in ER some researchers have proposed domain-specific numerical descriptors that describe each behavior β, to explicitly consider behavioral space during evolution as another criterion that helps guide the search. The justification for this is evident, given that the objective function is stated at a high-level of abstraction, then population management should also consider the behavioral features of the evolved solutions, that characterize them based on a high-level description of their performance. Therefore, researchers have proposed

diversity preservation techniques [34, 35] and open-ended search algorithms [15, 17]; Mouret and Doncieux [22] provide a comprehensive overview of previous works on behavioral evolution in ER.
In summary, a behavior should be understood as a higher-level description of program performance, compared to the semantics approach that employs a low-level description of performance. An individual's behavior is described in more general terms, accounting not only for program output, but also for the context in which the output was produced. For instance, for the SRS GP classifier described above, context is given by the SRS heuristic rule used to assign class label. Therefore, fitness, program semantics, and behavior can be understood as different levels of abstraction of the performance of a program. At one extreme, fitness provides a coarse look of performance; a single value (for each criteria) that attempts to capture a global evaluation. At the other end, semantics describe the performance of a program in great detail. On the other hand, behavioral descriptors move between fitness and semantics, providing a finer or coarser level of description, depending on how behaviors are meaningfully characterized within a particular domain.

3.1 Novelty Search

Following the behavior-based approach, Lehman and Stanley proposed the NS algorithm that eliminates an explicit objective function [15–17]. The search is not guided by a measure of *quality*, instead the selective pressure is provided by a measure of *uniqueness*. The strategy is to measure the amount of novelty each individual introduces into the search with respect to the progress the search has made at the moment at which the individual is created. Each solution is described by a domain dependent behavioral descriptor, where each individual is mapped to a single point in behavioral space, as described in the previous section.
A known limitation of fitness-based search is the tendency to converge and get trapped on local optima. A common solution to this problem is to incorporate niching or speciation techniques into an EA [19]. However, through the search for novelty diversity preservation introduces the sole selective pressure and can, in principle, avoid search stagnation.
In practice, NS uses a measure of local sparseness around each individual within behavioral space to estimate its novelty, considering the current population and novel solutions from previous generations. Therefore, the novelty measure is dynamic, since it can produce different results for the same individual depending on the population state and search progress. The proposed measure of sparseness ρ around each individual K described by its behavioral descriptor β, is given by

$$\rho(\beta) = \frac{1}{m}\sum_{i=0}^{m} dist(\beta, \alpha_i) \ , \tag{1}$$

where α_i is the ith-nearest neighbor in behavioral space of β with respect to the average distance $dist()$, which is a domain-dependent measure of behavioral difference between two descriptors. The number of neighbors m considered for the sparseness measure is an algorithm parameter. Given this definition, when the average distance is large, then the individual is within a sparse region of behavioral space, and it is in a dense region if the measure is small.
To compute sparseness, the original NS proposal is to consider the current population and an archive of novel individuals. An individual is added to the archive if

its sparseness is above a minimal threshold ρ_{min}, the second parameter of the NS algorithm. The archive can help the algorithm avoid backtracking, however it can grow large in size, and make calculating sparseness a computational bottleneck. Finally, at the conclusion of the run after a fixed number of generations, the best solution based on fitness (it is the only moment in which NS uses an explicit fitness function) from both the final population and the novelty archive is returned as the best solution found by the search.
Since its proposal in [15], and later works [16, 17], most applications of NS have focused on robotics, such as mobile robot navigation [15–17], morphology design [18] and gait control [17]. Only until recently has NS been used in general pattern recognition problems, particularly supervised classification [24] and unsupervised clustering [23]. This paper will use the work in [24] as the case study to analyze bloat in a NS-based GP search.

4 Novelty Search for Supervised Classification

This section describes the GP system used in this work to evolve supervised classifiers, and how NS is incorporated into the evolutionary process.

4.1 Static Range Selection GP Classifier

This work uses the Static Range Selection GP Classifier (SRS) described by Zhang and Smart [38]. In a classification problem, a pattern $\mathbf{x} \in \mathbb{R}^p$ has to be classified as belonging to a single class from $\Omega = \{\omega_1, ..., \omega_M\}$, where each ω_i represents a distinct class label. Then, in a supervised learning approach the goal is to build a mapping $g(\mathbf{x}) : \mathbb{R}^p \rightarrow \Omega$, that assigns each pattern $\mathbf{x}$ to a corresponding class ω_i, where g is derived based on evidence provided by a training set $\mathscr{T}$ of N p-dimensional patterns with a known classification. In this work, only two-class classification problems are considered. In SRS, $\mathbb{R}$ is divided into M non-overlapping regions, one for each class. Then, GP evolves a mapping $g(\mathbf{x}) : \mathbb{R}^p \rightarrow \mathbb{R}$, such that the region in $\mathbb{R}$ where pattern $\mathbf{x}$ is mapped to, determines the class to which it belongs. For a two-class problem, if $g(\mathbf{x}) > 0$ then $\mathbf{x}$ belongs to class ω_1, and belongs to ω_2 otherwise. The fitness function is simple, it consists on minimizing the classification error of g.

4.2 Novelty Search extension of SRS

As stated above, to apply NS with SRS the fitness function is substituted by the sparseness measure of Equation 1. Therefore, a proper domain specific behavioral descriptor must be proposed [8].
Accuracy Descriptor β^A: The training set $\mathscr{T}$ used by SRS-GPC contains sample patterns from each class. For a two-class problem with $\Omega = \{\omega_1, \omega_2\}$, If $\mathscr{T} = \{\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_L\}$, then the behavioral descriptor for each GP classifier K_i is a binary vector $\beta^{A_i} = (\beta_1, \beta_2, ..., \beta_L)$ of size L, where each vector element β_j is set to 1 if classifier K_i correctly classifies sample $\mathbf{y}_j$, and is set to 0 otherwise. In [24] an analysis on the β^A descriptor is given, considering the fitness landscape it produces.

Finally, given the proposed binary descriptors, a natural $dist()$ function for Equation 1 is the Hamming distance, that counts the number of bits that differ between two binary vectors. This similarity measure has been used to measure behavioral diversity in ER [22]. It is noteworthy to point out that [24] reports good performance on several synthetic classification problems. In particular, the authors report a clear trend, the performance of NS-based GP improves, relative to a control method, as problem difficulty increases; i.e., NS performs comparatively better on hard problems than on easy ones. The explanation for this observation is that, for easy problems random solutions can perform quite well. Therefore, the selective pressure provided by NS will not necessarily lead the search towards better solution in behavioral space, in fact the opposite might happen. Conversely, for difficult problems random solutions for a two-class problem will roughly produce 50% accuracy. Therefore, the diversity exploration of NS will have to lead the search towards better regions in the search space. In other words, when problem difficulty increases the search for novelty can lead towards quality.

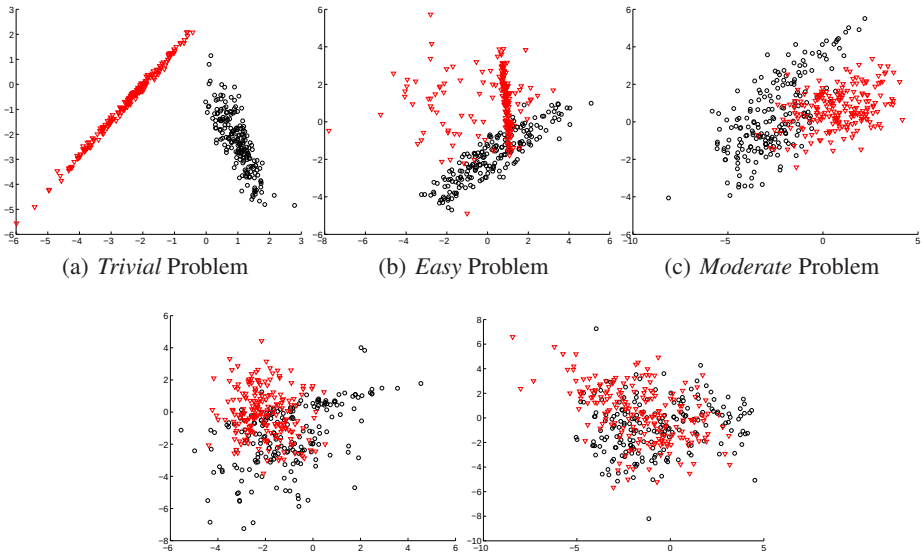
5 Experiments and Results

The goal of the experimental work is to test the NS-GP classifier on two-class classification problems, evaluating its performance based on classification error and the mean size of the evolved population, a good indicator of the effect bloating is having on a GP run [37]. The proposed algorithm will hereafter be denoted by NS-SRS. For comparative proposes, the basic SRS classifier is used as a control method.

Gaussian Mixture Models are used to generate five random synthetic problems, each with different amounts of class overlap and geometry. All problems are set in the $\mathbb{R}^2$ plane with $x, y \in [-10, 10]$, and 200 sample points were randomly generated for each class. The parameters for the GMM of each class were also randomly chosen, following the same strategy reported in [33]. The five problems are of increasing difficulty, denoted as: *Trivial*; *Easy*; *Moderate*; *Hard*; and *Hardest*; these problems are graphically depicted in Figure 1.

The parameters of both GP systems are given in Table 1. Moreover, for NS-SRS the NS parameters are set as follows: (1) the number of neighbors considered for sparseness computation is set to $m = 15$; and (2) the sparseness threshold is set to $\rho_{min} = 40$. Finally, all algorithms were coded using Matlab 2009a and the GPLAB toolbox [30].

Both algorithms are executed 30 times and performance is analyzed based on averages over all runs. For each problem, 200 sample points are created for each class, and 70% of the data is used for training and the rest for testing. In each run data partition is done independently at random. First, Table 2 presents the average classification error on the test data. The results are consistent with those reported in [24], NS-SRS performs well, with small performance differences compared to SRS. However, there is a trend, NS performs well on the more difficult problems, and worse on the easier ones. These results are similar to those reported in [24, 23], with similar conclusions. Basically, the explorative search performed by NS is fully exploited when random initial solution perform badly, under these conditions the search for novelty can lead towards better solutions. Conversely, for easy problems, random solution can perform quite well, thus the search for novelty can lead the search towards solutions with undesirable performance. However, this is



(a) *Trivial* Problem (b) *Easy* Problem (c) *Moderate* Problem

(d) *Hard* Problem (e) *Hardest* Problem

Fig. 1. Five synthetic 2-class problems used to evaluate each algorithm in ascending order of difficulty from left to right.

Table 1. Parameters for the GP systems.

Parameter	**Description**
Population size	100 individuals.
Generations	100 generations.
Initialization	*Ramped Half-and-Half*, with 6 levels of maximum depth.
Operator probabilities	Crossover $p_c = 0.8$, Mutation $p_\mu = 0.2$.
Function set	$\{+, -, \times, \div, \lvert\cdot\rvert, x^2, \sqrt{x}, log, sin, cos, if\}$.
Terminal set	$\{x_1,...,x_i,...,x_p\}$, where x_i is a dimension of the data patterns $\mathbf{x} \in \mathbb{R}^n$.
Hard maximum depth	20 levels.
Selection	Tournament of size 4.

not necessarily a limitation for NS-based systems, since for most real-world scenarios difficult problems should be expected.

Therefore, regarding fitness we can say that NS achieves basically equivalent performance compared with fitness-based search on the more interesting problems. Figures 2 and 3, on the other hand, show how the average size of the population evolves across generations. Consider SRS, Figure 2 shows typical GP behavior, with a clear tendency of code growth across generations; i.e., SRS bloats like any fitness-based GP search. Second, in the case of NS-SRS, code growth is analyzed from three different perspectives. First, considering how the average size of the population grows across generations, shown in Figure 3(a). Second, considering how the average size of the individuals in the archive grows, shown in Figure 3(b). Finally, when both the archive and the population are considered concur-

Table 2. Average classification error and standard error of the best solution found by each algorithm.

Problem	SRS Ave.	SRS Std.	NS-SRS Ave.	NS-SRS Std.
Trivial	0.004	0.007	0.007	0.008
Easy	0.105	0.040	0.144	0.044
Moderate	0.136	0.033	0.159	0.041
Hard	0.260	0.052	0.266	0.053
Hardest	0.365	0.033	0.370	0.043

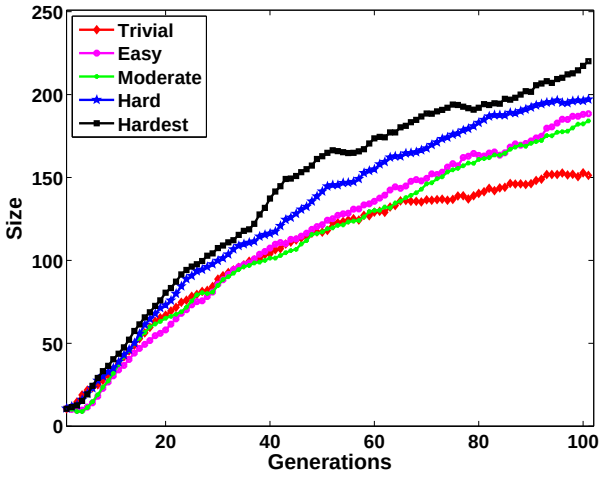
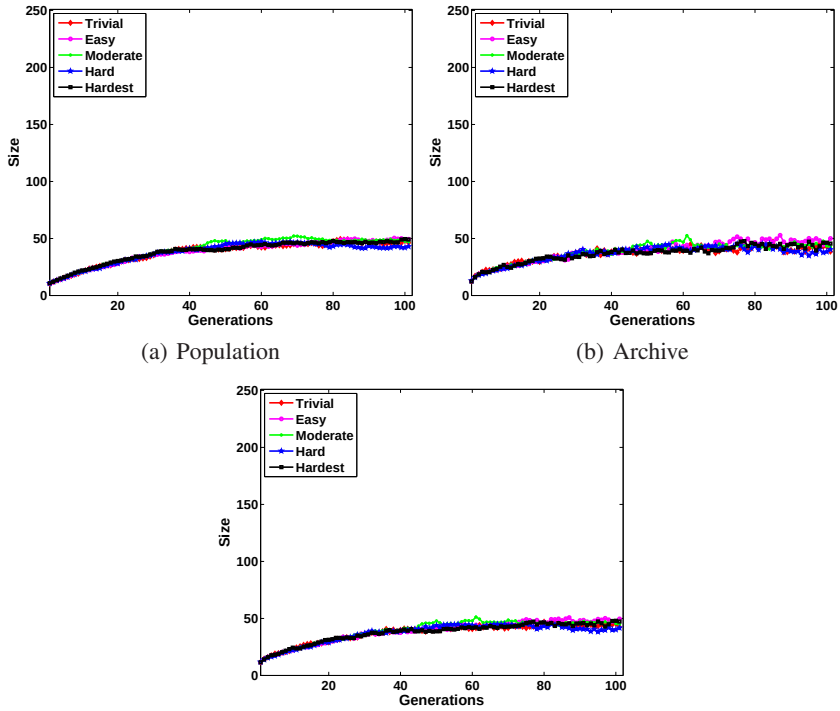


Fig. 2. Evolution of tree size for SRS on each problem; curves represent the average size over thirty runs.

rently in each generation, this is shown in Figure 3(c). In all cases, it is clear that NS controls code growth quite effectively. A numerical comparison is given in Table 3, with the average population size in the final generation. It is clear that program size is considerably larger with fitness-based search, without the quality of the results being compromised. There are some additional observations that are of interest in the results of Table 3. It seems that standard GP bloats more as problem difficulty increases, which is coherent with the FCBT. On the other hand, NS shows the same average program size for all problems. Moreover, average program size in the archive is slightly smaller than the average program size in the population; i.e., archived individuals are smaller than the population average. This is noteworthy, because it suggests that novel individuals tend to be smaller, an unexpected result.

6 Summary and Conclusions

Bloat is one of the main research topics in modern GP literature, and several theories have been developed that attempt to explain its underlying causes. Moreover, given the detrimental effect that bloat has on GP runs, a variety of bloat control



(a) Population (b) Archive

(c) Population + Archive

Fig. 3. Evolution of tree size for NS-SRS on each problem; curves represent the average size over thirty runs. (a) Considers individuals in the population; (b) considers individuals in the archive; and (c) considers all of the individuals at each generation.

Table 3. Average program size in the final generation for each algorithm. For the NS algorithm, the population (Pop), archive (Arc) and both (Pop+Arc) are considered.

Problem	SRS	NS-SRS Pop	NS-SRS Arc	NS-SRS Pop+Arc
Trivial	151.3	46.17	38.57	42.37
Easy	188.5	49.25	49.89	49.57
Moderate	184	47.04	42.54	44.79
Hard	197	42.93	40.17	41.55
Hardest	220.1	49.05	45.52	47.29

methods have been proposed. Nonetheless, a complete explanation and a general effective strategy have not been devised. A general agreement, however, is that the search for fitness is the one core element that leads towards bloated populations in GP; which is explicitly stated by the fitness-causes-bloat theory. If this is so, then bloat would seem to be unavoidable, since fitness based is a requirement of most evolutionary algorithms.

This work, on the other hand, studies bloat in a GP system where fitness is not considered explicitly. The proposal is to evolve GP programs using novelty

search, where fitness is substituted my a measure of program uniqueness. Initial results are indeed encouraging, it seems that bloat is eliminated, or at least controlled, by the evolutionary dynamics induced by NS. Experimental tests seem to confirm the FCBT, by showing that when an explicit fitness is abandoned then the evolving population will not bloat. However, these results are preliminary and much work still needs to be done. A complete and comprehensive study on bloat and behavior-based search is still necessary, but this work provides a first approximation at a plausible approach for bloat-free GP.

Acknowledgments Funding for this work provided by CONACYT (Mexico) Basic Science Research Project No. 178323 and DGEST (Mexico) Research Project No. TIJ-ING-2012-110. Second and third author are supported by CONACYT (Mexico) scholarships, respectively scholarships No. 232288 and No. 226981.

References

1. L. Beadle and C. Johnson. Semantically driven crossover in genetic programming. In *Proceedings of the Tenth Conference on Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, CEC'08, pages 111–116. IEEE Press, 2008.
2. L. Beadle and C. G. John'son. Semantically driven mutation in genetic programming. In *Proceedings of the Eleventh Conference on Congress on Evolutionary Computation*, CEC'09, pages 1336–1342. IEEE Press, 2009.
3. R. A. Brooks. *Cambrian intelligence: the early history of the new AI*. MIT Press, Cambridge, MA, USA, 1999.
4. R. Dawkins. *Climbing Mount Improbable*. W.W. Norton & Company, 1996.
5. S. Dignum and R. Poli. Generalisation of the limiting distribution of program sizes in tree-based genetic programming and analysis of its effects on bloat. In *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, GECCO '07, pages 1588–1595, New York, NY, USA, 2007. ACM.
6. M. Garcia, L. Trujillo, F. Fernández-de Vega, J. Merelo, and G. Olague. Evospace-interactive: A framework to develop distributed collaborative-interactive evolutionary algorithms for artistic design. In P. Machado and J. McDermott, editors, *Proceedings from the 2nd International Conference on Evolutionary and Biologically Inspired Music, Sound, Art and Design, EvoMUSART '12*, LNCS, pages 121–132. Springer-Verlag, 2013.
7. R. Harper. Spatial co-evolution: quicker, fitter and less bloated. In *Proceedings of the fourteenth international conference on Genetic and evolutionary computation conference*, GECCO '12, pages 759–766, New York, NY, USA, 2012. ACM.
8. S. Kistemaker and S. Whiteson. Critical factors in the performance of novelty search. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, GECCO '11, pages 965–972. ACM, 2011.
9. T. Kowaliw, A. Dorin, and J. McCormack. Promoting creative design in interactive evolutionary computation. *Evolutionary Computation, IEEE Transactions on*, 16(4):523 –536, 2012.

10. J. Koza. Human-competitive results produced by genetic programming. *Genetic Programming and Evolvable Machines*, 11(3):251–284, 2010.
11. J. R. Koza. *Genetic programming: on the programming of computers by means of natural selection*. MIT Press, Cambridge, MA, USA, 1992.
12. K. Krawiec and T. Pawlak. Locally geometric semantic crossover: a study on the roles of semantics and homology in recombination operators. *Genetic Programming and Evolvable Machines*, 14(1):31–63, 2013.
13. W. B. Langdon and R. Poli. Fitness causes bloat. In *Proceedings of the Second On-line World Conference on Soft Computing in Engineering Design and Manufacturing*, pages 13–22, 1997. Springer-Verlag.
14. W. B. Langdon and R. Poli. Fitness causes bloat: Mutation. In *Proceedings of the First European Workshop on Genetic Programming*, EuroGP '98, pages 37–48, London, UK, UK, 1998. Springer-Verlag.
15. J. Lehman and K. O. Stanley. Exploiting open-endedness to solve problems through the search for novelty. In *Proceedings of the Eleventh International Conference on Artificial Life, Cambridge, MA*, ALIFE XI. MIT Press, 2008.
16. J. Lehman and K. O. Stanley. Efficiently evolving programs through the search for novelty. In *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, GECCO '10, pages 837–844. ACM, 2010.
17. J. Lehman and K. O. Stanley. Abandoning objectives: Evolution through the search for novelty alone. *Evol. Comput.*, 19(2):189–223, 2011.
18. J. Lehman and K. O. Stanley. Evolving a diversity of virtual creatures through novelty search and local competition. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, GECCO '11, pages 211–218. ACM, 2011.
19. S. W. Mahfoud. *Niching methods for genetic algorithms*. PhD thesis, Champaign, IL, USA, 1995. UMI Order No. GAX95-43663.
20. N. F. McPhee, B. Ohs, and T. Hutchison. Semantic building blocks in genetic programming. In *Proceedings of the 11th European conference on Genetic programming*, EuroGP'08, pages 134–145, Berlin, Heidelberg, 2008. Springer-Verlag.
21. A. Moraglio, K. Krawiec, and C. G. Johnson. Geometric semantic genetic programming. In *Proceedings of the 12th international conference on Parallel Problem Solving from Nature - Volume Part I*, PPSN'12, pages 21–31, Berlin, Heidelberg, 2012. Springer-Verlag.
22. J. B. Mouret and S. Doncieux. Encouraging behavioral diversity in evolutionary robotics: An empirical study. *Evol. Comput.*, 20(1):91–133, 2012.
23. E. Naredo and L. Trujillo. Searching for novel clustering programs. To appear in *Proceeding from the Genetic and Evolutionary Computation Conference, GECCO 2013*. ACM, 2013.
24. E. Naredo, L. Trujillo, and Y. Martínez. Searching for novel classifiers. In *Proceedings from the 16th European Conference on Genetic Programming, EuroGP 2013*, volume 7831 of *LNCS*, pages 145–156. Springer-Verlag, 2013.
25. A. L. Nelson, G. J. Barlow, and L. Doitsidis. Fitness functions in evolutionary robotics: A survey and analysis. *Robot. Auton. Syst.*, 57(4):345–370, 2009.
26. S. Nolfi and D. Floreano. *Evolutionary Robotics: The Biology,Intelligence,and Technology*. MIT Press, Cambridge, MA, USA, 2000.

27. C. Ofria and C. O. Wilke. Avida: a software platform for research in computational evolutionary biology. *Artif. Life*, 10(2):191–229, 2004.
28. R. Poli, W. B. Langdon, and S. Dignum. On the limiting distribution of program sizes in tree-based genetic programming. In *Proceedings of the 10th European conference on Genetic programming*, EuroGP'07, pages 193–204, Berlin, Heidelberg, 2007. Springer-Verlag.
29. S. Silva. Reassembling operator equalisation: a secret revealed. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, GECCO '11, pages 1395–1402, New York, NY, USA, 2011. ACM.
30. S. Silva and J. Almeida. Gplab–a genetic programming toolbox for matlab. In L. Gregersen, editor, *Proceedings of the Nordic MATLAB conference*, pages 273–278, 2003.
31. S. Silva and E. Costa. Dynamic limits for bloat control in genetic programming and a review of past and current bloat theories. *Genetic Programming and Evolvable Machines*, 10(2):141–179, 2009.
32. S. Silva, S. Dignum, and L. Vanneschi. Operator equalisation for bloat free genetic programming and a survey of bloat control methods. *Genetic Programming and Evolvable Machines*, 13(2):197–238, 2012.
33. L. Trujillo, Y. Martínez, E. Galván-López, and P. Legrand. Predicting problem difficulty for genetic programming applied to data classification. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, GECCO '11, pages 1355–1362, New York, NY, USA, 2011. ACM.
34. L. Trujillo, G. Olague, E. Lutton, and F. F. De Vega. Discovering several robot behaviors through speciation. In *Proceedings of the 2008 conference on Applications of evolutionary computing*, Evo'08, pages 164–174. Springer-Verlag, 2008.
35. L. Trujillo, G. Olague, E. Lutton, F. Fernández de Vega, L. Dozal, and E. Clemente. Speciation in behavioral space for evolutionary robotics. *Journal of Intelligent & Robotic Systems*, 64(3-4):323–351, 2011.
36. N. Q. Uy, N. X. Hoai, M. O'Neill, R. I. Mckay, and E. Galván-López. Semantically-based crossover in genetic programming: application to real-valued symbolic regression. *Genetic Programming and Evolvable Machines*, 12(2):91–119, 2011.
37. L. Vanneschi, M. Castelli, and S. Silva. Measuring bloat, overfitting and functional complexity in genetic programming. In *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, GECCO '10, pages 877–884, New York, NY, USA, 2010. ACM.
38. M. Zhang and W. Smart. Using gaussian distribution to construct fitness functions in genetic programming for multiclass object classification. *Pattern Recogn. Lett.*, 27(11):1266–1274, 2006.