Searching for Novel Classifiers

Enrique Naredo, Leonardo Trujillo*, 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
enriquenaredo@gmail.com, leonardo.trujillo@tectijuana.edu.mx, ysaraimr@gmail.com
home page: https://sites.google.com/site/leonardotrujillogp

Abstract. Natural evolution is an open-ended search process without an a priori fitness function that needs to be optimized. On the other hand, evolutionary algorithms (EAs) rely on a clear and quantitative objective. The Novelty Search algorithm (NS) substitutes fitness-based selection with a *novelty* criteria; i.e., individuals are chosen based on their uniqueness. To do so, individuals are described by the behaviors they exhibit, instead of their phenotype or genetic content. NS has mostly been used in evolutionary robotics, where the concept of behavioral space can be clearly defined. Instead, this work applies NS to a more general problem domain, classification. To this end, two behavioral descriptors are proposed, each describing a classifier's performance from two different perspectives. Experimental results show that NS-based search can be used to derive effective classifiers. In particular, NS is best suited to solve difficult problems, where exploration needs to be encouraged and maintained.

Keywords: Novelty Search, Classification, Genetic Programming

1 Introduction

Research in Evolutionary Computation (EC) has produced search and optimization algorithms that frequently achieve promising new results in diverse domains [4]. Therefore, the practical value of evolutionary algorithms (EAs) is by now widely accepted. Nonetheless, for some within the field a conceptual, or even philosophical, problem remains regarding most EAs. At their core, EAs are simple abstractions of Neo-Darwinian evolution. However, instead of the open-ended nature of biological evolution, EAs are objective driven, just like any conventional optimization algorithm. Therefore, EAs are expected to converge on a small subset of local optima within a static fitness landscape.

This difference, however, is not a general one. In fact, some of the earliest EAs were open-ended techniques [1]. While such EAs are still abstract simplifications of evolution, they do integrate an open-ended feature not present in most standard algorithms. Open-ended algorithms have mostly been used in

*Corresponding author.

specialized domains, such as artificial life environments [10] and interactive search [3]. Only recently have open-ended algorithms been proposed to solve mainstream problems. In particular, Lehman and Stanley [5–7] proposed novelty search (NS), an EA where the objective function is abandoned. Instead, selective pressure considers the novelty, or "uniqueness", of each individual by describing its behavior. Thus, fitness in NS is implicitly captured within the behavioral description of each individual. While such an approach might seem counterintuitive, experimental results are promising and show that highly fit solutions can emerge from a search that does not consider fitness explicitly.

Despite the success of NS, it might also appear to be a niche strategy that is well-suited for a small subset of domains. This paper explores the usefulness of NS on a common type of problem: classification. The core element of NS is that each individual is described by a behavioral descriptor, which is then used to measure the novelty that each solution introduces into the search. This paper proposes two behavioral descriptors for a Genetic Programming (GP) classifier. Each descriptor introduces a different behavioral space and corresponding fitness landscapes. Experimental results are encouraging, NS-based classification achieves competitive results compared to a canonical GP. Moreover, the paper analyzes some of the practical considerations that must be accounted for if NS is to be used successfully.

The paper is organized as follows. First, Section 2 describes the NS algorithm. Afterwards, Section 3 presents the proposed NS-based GP algorithm for data classification and two behavioral descriptors for evolved classifiers. Then, Section 4 presents the experiments and an analysis of the results. Finally, a summary of the paper and concluding remarks are given in Section 5.

2 Novelty Search

The main idea behind NS is to eliminate the objective function from a search [5–7]. In other words, evolution is not guided by the measured *quality* of each individual, instead it is guided by a measure of *uniqueness*; i.e., how novel each individual is with respect to what has been found earlier by the search. A known limitation of the traditional objective-based search is a tendency to converge and stagnate on local optima, particularly in multi-modal problems with irregular fitness landscapes. Within EC, diversity preservation techniques are usually incorporated within an EA to overcome the above shortcoming. However, most proposals can be regarded as ad-hoc solutions that must continuously attempt to balance exploration and exploitation during the search . Conversely, through the search for novelty alone, diversity preservation introduces the sole selective pressure during the search.

NS operates based on the concept of behavior, where each individual is described based on the functional behavior it exhibits. Therefore, individuals are described in behavioral space, instead of the more common genotypic, phenotypic or fitness spaces that are used for diversity preservation [13, 14]. Behaviors are expressed by a domain dependent descriptor, such that each individual is

mapped to a single point in behavioral space. A behavior implicitly represents the fitness of an individual, providing a fine grained view of its performance or just a different domain specific perspective. Since many individuals in genotypic space express the same behavior, and are thus mapped to the same point in behavioral space, the search for novelty is often feasible. Lehman and Stanley argue that since the number of simple behaviors for any given problem is relatively small, then the search for novelty must necessarily lead to more functionally complex solutions. The concept of behavior as described above is closely related to the concept of semantics in GP [15].

In summary, NS uses a measure of *novelty* to characterize each individual. More precisely, the sparseness of each individual within behavioral space is measured, with respect to other individuals within the population and novel solutions from previous generations. An important observation is that such a measure of novelty is dynamic; i.e., it can produce different results for the same individual depending on the population state and search progress at a given generation. NS measures the sparseness ρ around each individual i, described by its behavioral descriptor $\mathbf{x}$, using the average distance to the k-nearest neighbors in behavioral space, with k an algorithm parameter, given by

$$\rho(\mathbf{x}) = \frac{1}{k} \sum_{i=0}^{k} dist(\mathbf{x}, \mu_i) \ , \tag{1}$$

where μ_i is the ith-nearest neighbor of $\mathbf{x}$ with respect to distance metric $dist$, a domain-dependent measure of behavioral difference between two descriptors. If the average distance is large then the individual lies within a sparse region of behavioral space and it lies in a dense region when the measure is small.

In NS, sparseness is computed based on the contents of the current population and an archive of individuals that at one moment were considered to be novel. Therefore, 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 also mitigate backtracking by the search process. This can also be seen as a shortcoming of the approach, since if the archive grows then a higher computational cost is incurred to compute sparseness. To address this problem, [6] implements the archive as a fixed size FIFO queue.

The NS algorithm provides an open-ended evolutionary approach to solve mainstream scientific and engineering problems. However, since its proposal in [5], and later works [6, 7], most applications of NS have focused on robotics, [5–8]. All of these works are part of a wider area of research known as evolutionary robotics (ER), where evolutionary algorithms are used to solve problems related to robot design and control. Within ER, the topic of evolving a diverse set of behaviors has also been addressed in other ways. For instance, [13, 14] propose to integrate speciation techniques to evolve a diverse set of robot behaviors. A good review on this topic is given by Mouret and Doncieux [9]. Search algorithms that explicitly contemplate behaviors seem well suited for robotics, since most high-level tasks can usually be solved in structurally different ways, guaranteeing multi-modal search spaces.

On the other hand, NS has not been used in most domains. A noteworthy exception is [16], where NS is integrated with an interactive evolutionary system. To the authors knowledge, however, applying NS to mainstream problems is not yet common. The present work proposes the use of NS for a ubiquitous pattern analysis problem, data classification.

3 Classification with Novelty Search

This section presents the proposed behavioral descriptors for GP-based classifiers and discusses the fitness landscape of each.

3.1 Static Range Selection GP Classifier

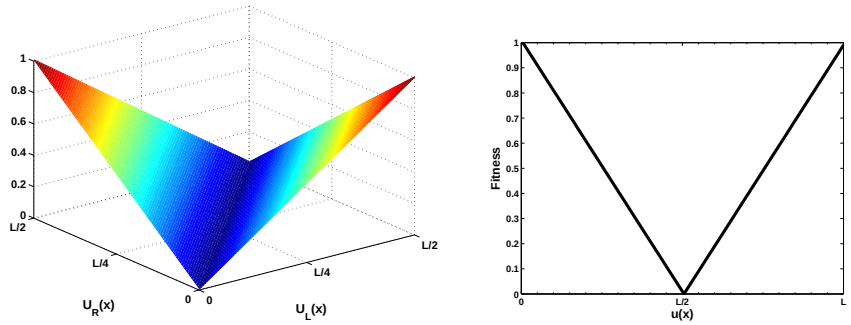
This work uses the Static Range Selection GP Classifier (SRS-GPC) described by Zhang and Smart [17]. In a classification problem, a pattern $\mathbf{x} \in \mathbb{R}^P$ has to be classified as belonging to a single class from $\Omega = \{\omega_1, ..., \omega_M\}$, where each ω_i represents a distinct 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-GPC, $\mathbb{R}$ 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 maximizing the classification accuracy of g.

3.2 Novelty Search extension of SRS-GPC

As stated above, to apply NS with SRS-GPC the fitness function is substituted by the sparseness measure of Equation 1. Therefore, a proper domain specific behavioral descriptor must be proposed [2]. Two descriptors are proposed next, each inducing a different fitness landscape and behavioral neighborhoods.

Class Descriptor (CD): The training set $\mathscr{T}$ used by SRS-GPC contains sample patterns from each class. Then, for a two-class problem with $\Omega = \{\omega_1, \omega_2\}$ the CD is constructed in the following way. 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 $\mathbf{a_i} = (a_1, a_2, ...a_L)$ of size L, where each vector element a_j is set to 1 if classifier K_i assigns label ω_1 to pattern $\mathbf{y}_j$ and is set to 0 otherwise.

Accuracy Descriptor (AD): The second descriptor considers the accuracy of a classifier at a fine scale. 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 $\mathbf{b_i} = (b_1, b_2, ...b_L)$ of size L, where each vector element b_j is set to 1 if classifier K_i correctly classifies sample $\mathbf{y}_j$ and is set to 0 otherwise.



(a) *Class Descriptor* (b) *Accuracy Descriptor*

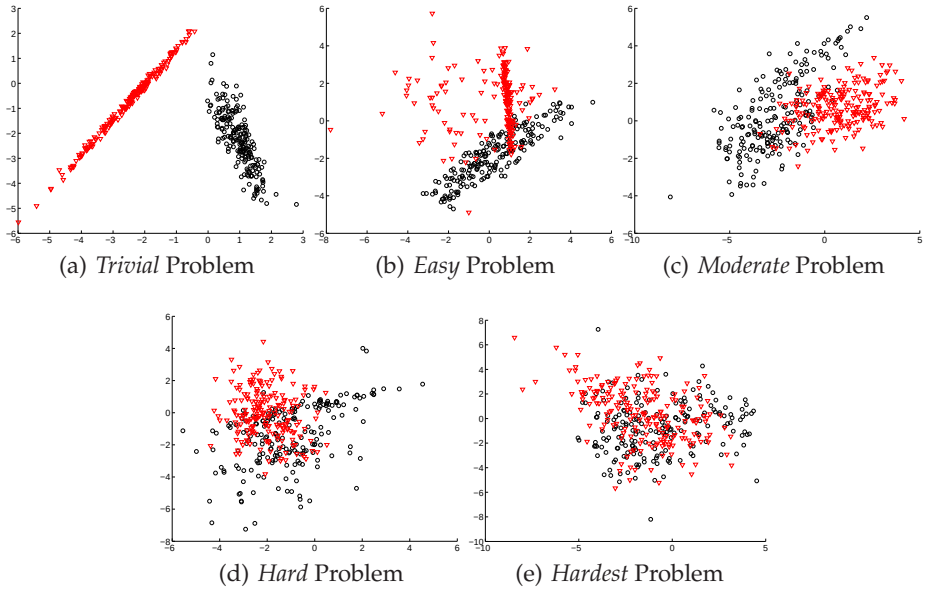
Fig. 1. Fitness landscape in behavioral space for each descriptor.

While both descriptors are binary vectors of size L, each induces a different fitness landscape in behavioral space. Suppose that the number of training examples from each class is $\frac{L}{2}$, and suppose that they are ordered in such a way that the first $\frac{L}{2}$ elements in $\mathscr{T}$ correspond to class label ω_1. Let $\mathbf{x}$ represent a binary vector, and function $u(\mathbf{x})$ return the number of 1s in $\mathbf{x}$. Moreover, let K_O be the *optimal* classifier that achieves a perfect accuracy on the training set.

Then, the CD of K_O is given by $\mathbf{a}^1 = (1_1, 1_2, ...1_{\frac{L}{2}}, 0_{\frac{L}{2}+1}....0_L)$. The AD of K_O is given by $\mathbf{b}^1$ where $u(\mathbf{b}^1) = L$. Moreover, for a two-class problem, an equally useful solution is to take the opposite (complement) behaviors and invert the classification, such that a 1 is converted to a 0 and vice-versa. These mirror behaviors are $\mathbf{a}^0 = (0_1, 0_2, ...0_{\frac{L}{2}}, 1_{\frac{L}{2}+1}....1_L)$ for the CD and $\mathbf{b}^0$ with $u(\mathbf{b}^0) = 0$ for the AD. The fitness landscapes in behavioral space are depicted in Figure 1.

For a two-class problem with a reasonable degree of difficulty, the initial generations of a GP search should be expected to contain close to random classifiers, with roughly a 50% accuracy. For the CD descriptor, behavioral space is organized on a two dimensional surface, such that one axis u_L considers the number of ones on the left hand side, first $\frac{L}{2}$ bits, of a behavior descriptor $\mathbf{a}$, and u_R considers the remaining $\frac{L}{2}$ bits; see Figure 1(a). Notice that the middle valley of the fitness landscape corresponds to random classifiers, with worst case performance. Hence, NS will push the search towards either of the two global optima, $\mathbf{a}^1$ and $\mathbf{a}^0$ On the other hand, for the AD descriptor, early behaviors will mostly exhibit descriptors with equal proportions of zeros and ones; see Figure 1(b). Then, NS will progressively explore towards two opposite points in behavioral space, $\mathbf{b}^1$ or $\mathbf{b}^0$. The effect on performance of these differences, between the CD and the AD, are explored experimentally in the following section.

Finally, given the above binary descriptors, a natural $dist()$ function for Equation 1 is the Hamming distance, that counts the number of bits that differ be-



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

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

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

tween two binary vectors. This similarity measure has been used to measure behavioral diversity in ER [9].

4 Experiments

The performance of the NS-based GP classifier is examined. Several different versions are tested and compared. First, the basic SRS-GPC classifier. Second, the NS variant with CD in two different versions. One configuration uses a novelty archive of unbounded size, while the second one used a FIFO archive with limited size as in [6]. Hereafter, the former is referred to as NS-CD and the latter as NS-CD-L. Similarly, two NS variants with AD are tested, NS-AD and NS-AD-L.

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 [12]. The five problems are of increasing difficulty, denoted as: *Trivial*; *Easy*; *Moderate*; *Hard*; and *Hardest*; these problems are graphically depicted in Figure 2.

As stated above, five different algorithms are experimentally compared: SRS-GPC, NS-CD, NS-CD-L, NS-AD and NS-AD-L. All algorithms share the same GP representation and genetic operators, a tree-based Koza style algo-

Table 1. Parameters for the GP-based search.

Parameter	Description
Population size	200 individuals.
Generations	200 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 $\mathbf{x} \in \mathbb{R}^n$.
Bloat control	Dynamic depth control.
Initial dynamic depth	6 levels.
Hard maximum depth	20 levels.
Selection	Tournament.

rithm with subtree mutation and crossover. The parameters shared by all algorithms are summarized in Table 1. Additionally, SRS-GPC also uses a keep-best elitism strategy.

For the NS-based algorithms two different parameter settings are used. In particular, two different values for the archive threshold ρ_{min} are used, 40 and 80. Parameter k is set to 15 for all algorithms. Finally, all algorithms were coded using Matlab 2009a and the GPLAB toolbox [11].

For each algorithm, 30 different runs were executed for each problem shown in Figure 2. In each run, the data set is randomly dividing into training and testing sets, with the former containing 70% of the data samples.

First, tables 2 and 3 compare the performance of every algorithm on each problem, considering the test data from each run and presenting the average classification error $\pm$ the standard error. In Table 2 the NS-based algorithms use $k = 15$ and $\rho_{min} = 80$, while in Table 3 $k = 15$ and $\rho_{min} = 40$. To verify statistical significance, the Wilcoxon rank-sum test is performed between the control algorithm SRS-GPC and each of the NS algorithms. In tables 3 and 2 an asterisk indicates that the corresponding NS-algorithm achieves statistically equivalent results with SRS-GPC at the $\alpha = 0.05$ significance level. In general, these results show that AD produces better performance than CD and that limiting the size of the archive does not affect performance, and in some cases improves it. Additionally, a lower ρ_{min} encourages better performance in most algorithms. Moreover, with respect to each problem we can state the following. First, for the trivial problem, all of the algorithms can solve it nearly perfectly. Second, for the *easy* problem NS produces slightly worse results than standard search. However, both problems are quite easy, far from the type of data generally encountered in real-world scenarios. Finally, for the *moderate* and *hard* problems, the NS-algorithm achieves equal performance with respect to SRS-GPC.

Figures 3-10 examines the evolution of the NS-based algorithms. Each figure contains two plots that show averages over all runs; these are: (1) evolution of fitness and (2) evolution of sparseness. First, with respect to the fitness of the best solution at each generation, the difference in performance between each problem is evident and consistent across all algorithms. The second plot in each figure shows the sparseness value associated to the best solution at each gener-

Table 2. Average classification error and standard error of the best solution found by each algorithm on each problem; NS-based algorithms use $k = 15$ and $\rho_{min} = 80$.

Problem	SRS-GPC	NS-CD	NS-CD-L	NS-AD	NS-AD-L
Trivial	0.005 ± 0.006	$0.006 \pm 0.008^*$	$0.006 \pm 0.010^*$	$0.002 \pm 0.005^*$	$0.006 \pm 0.007^*$
Easy	0.080 ± 0.026	0.131 ± 0.035	0.128 ± 0.031	0.115 ± 0.034	0.136 ± 0.033
Moderate	0.129 ± 0.030	0.150 ± 0.030	$0.132 \pm 0.041^*$	$0.152 \pm 0.050^*$	$0.133 \pm 0.041^*$
Hard	0.255 ± 0.049	$0.279 \pm 0.044^*$	0.287 ± 0.039	0.282 ± 0.044	$0.272 \pm 0.057^*$
Hardest	0.374 ± 0.048	$0.342 \pm 0.037^*$	$0.381 \pm 0.053^*$	$0.367 \pm 0.049^*$	$0.380 \pm 0.045^*$

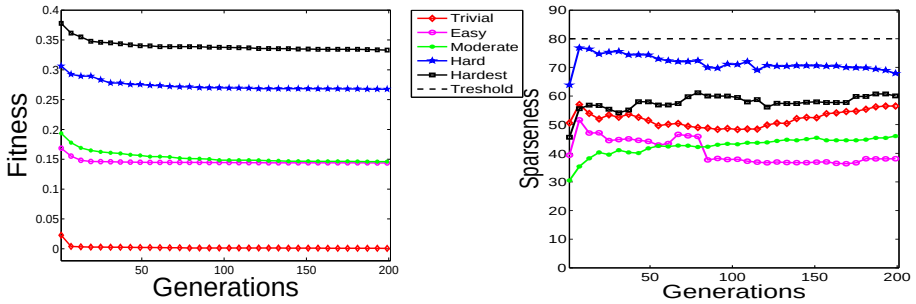
Table 3. Average classification error and standard error of the best solution found by each algorithm on each problem; NS-based algorithms use $k = 15$ and $\rho_{min} = 40$.

Problem	SRS-GPC	NS-CD	NS-CD-L	NS-AD	NS-AD-L
Trivial	0.005 ± 0.006	$0.004 \pm 0.006^*$	$0.001 \pm 0.004^*$	$0.005 \pm 0.006^*$	$0.005 \pm 0.007^*$
Easy	0.080 ± 0.026	0.124 ± 0.032	0.152 ± 0.124	0.130 ± 0.034	0.134 ± 0.037
Moderate	0.129 ± 0.030	0.153 ± 0.045	$0.180 \pm 0.146^*$	$0.148 \pm 0.044^*$	$0.149 \pm 0.036^*$
Hard	0.255 ± 0.049	$0.281 \pm 0.051^*$	$0.330 \pm 0.155^*$	$0.271 \pm 0.053^*$	$0.271 \pm 0.053^*$
Hardest	0.374 ± 0.048	$0.383 \pm 0.045^*$	$0.406 \pm 0.111^*$	$0.385 \pm 0.050^*$	$0.365 \pm 0.037^*$

ation. A horizontal line in these plots shows the corresponding threshold value, set to 80 in figures 3-6 and set to 40 in figures 7-10. In the former group, on average, the best solution does not reach the threshold. This exhibits the importance of ρ_{min}, if it is not set correctly then the best solution might not be saved in the archive; thus explaining the overall worse performance shown in Table 2. It is apparent that ρ_{min} should be lower, as is the case in figures 7-10. Nonetheless, with $\rho_{min} = 40$ the sparseness value of the best individual rises above the threshold only on the more difficult problems. This illustrates the main assumption behind the usefulness of NS, that random solutions will mostly exhibit bad fitness, and thus good solutions will tend to also be novel ones. Nonetheless, even if the best solution at each generation is not incorporated into the archive, it appears that sufficiently good solutions are saved, based on the test performance summarized in tables 2 and 3, that is mostly equivalent to the standard GP search.

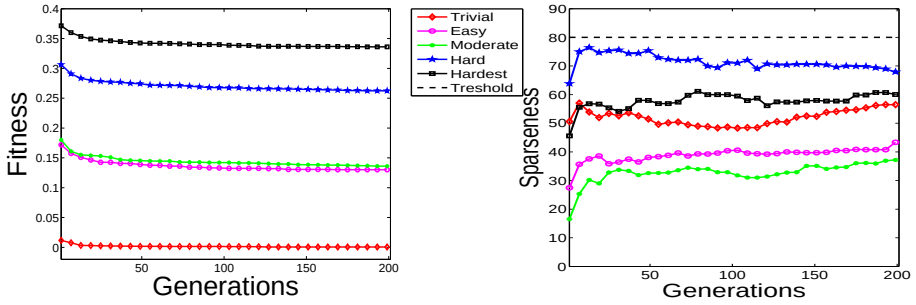
5 Conclusions

This paper uses a GP system based on the NS algorithm to search for data classifiers. To the authors knowledge, the work represents the first attempt to leverage NS to solve a common problem in pattern analysis and recognition, since previous applications of NS were primarily focused on robotic tasks. This line of research follows other recent works where solution behavior [14, 9], or solution semantics [15], are explicitly considered during a population-based search. To do so, two domain-specific behavioral descriptors were proposed, the Class Descriptor and the Accuracy Descriptor. In general, both descriptors appear to produce equivalent performance, in most cases statistically similar to a canon-



(a) *Fitness* (b) *Sparseness*

Fig. 3. Evolution of NS-CD with parameters $k = 15$ and $\rho_{min} = 80$.

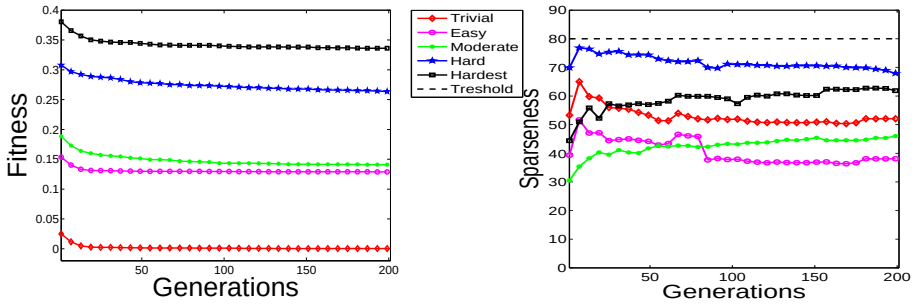


(a) *Fitness* (b) *Sparseness*

Fig. 4. Evolution of NS-CD-L with parameters $k = 15$ and $\rho_{min} = 80$.

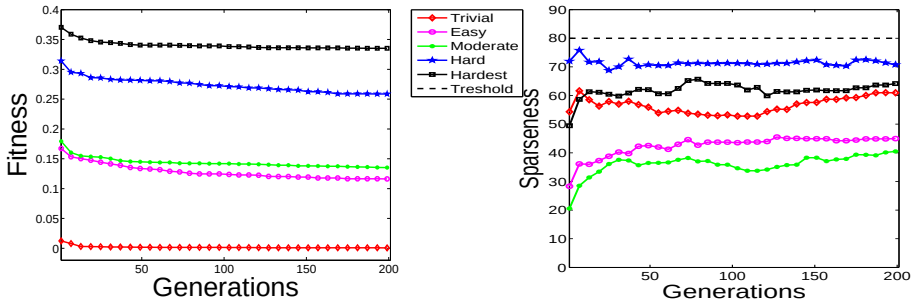
ical GP search. Moreover, it appears that NS-based search exhibits the best results when confronted with difficult problems. It seems that the reason for this is that generating a high-quality solution at random is less probable for difficult problems, then the incentive for behavioral exploration is incremented and the search for novelty will indeed lead towards quality during the search. For simple problems, however, the explorative capacity of NS is mostly unexploited or even a detriment to the search; i.e., if random solutions have a high fitness then novelty could easily lead the search towards worse results. Finally, while both descriptors, AD and CD, achieve similar performance on these tests, their differences must be studied and exploited further. In particular, the AD descriptor can only be used in supervised learning problems since it assumes knowledge of a ground truth set or classified samples. The CD descriptor, however, is less restrictive in this sense. Therefore, future work will center on exploring the usefulness of NS on the more difficult problem of non-supervised learning.

Acknowledgments. This research was supported by CONACYT (Mexico) Basic Science Research Grant No. 178323, "*Predicción de Rendimiento y Dificultad de*



(a) *Fitness* (b) *Sparseness*

Fig. 5. Evolution of NS-AD with parameters $k = 15$ and $\rho_{min} = 80$.



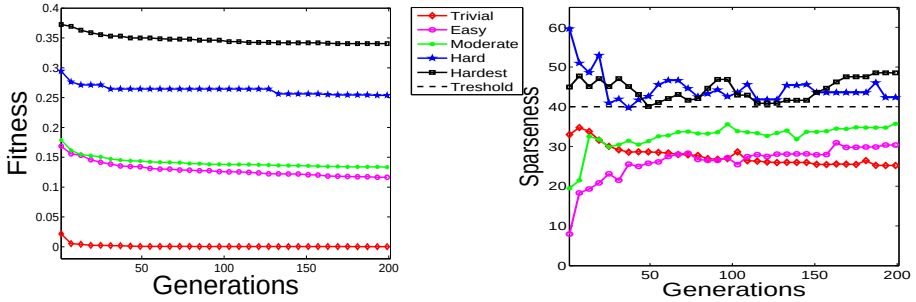
(a) *Fitness* (b) *Sparseness*

Fig. 6. Evolution of NS-AD-L with parameters $k = 15$ and $\rho_{min} = 80$.

Problemas en Programación Genética". First and third authors were supported by PRONABES-DGEST (Mexico) scholarships, respectively No. 20120000634 and No. 20120000735.

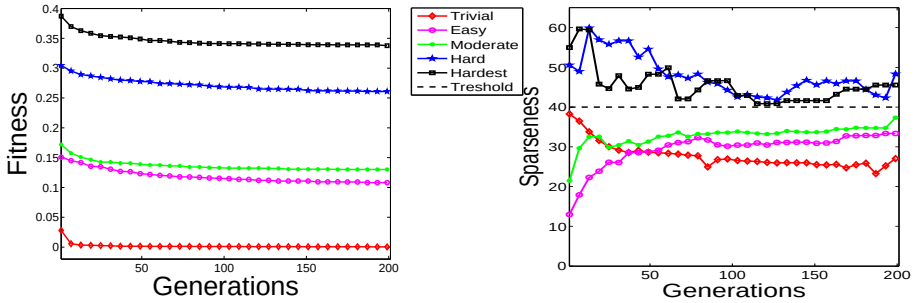
References

1. R. Dawkins. *Climbing Mount Improbable*. W.W. Norton & Company, 1996.
2. 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.
3. T. Kowaliw, A. Dorin, and J. McCormack. Promoting creative design in interactive evolutionary computation. *Evolutionary Computation, IEEE Transactions on*, 16(4):523 –536, 2012.
4. J. Koza. Human-competitive results produced by genetic programming. *Genetic Programming and Evolvable Machines*, 11(3):251–284, 2010.
5. 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.



(a) *Fitness* (b) *Sparseness*

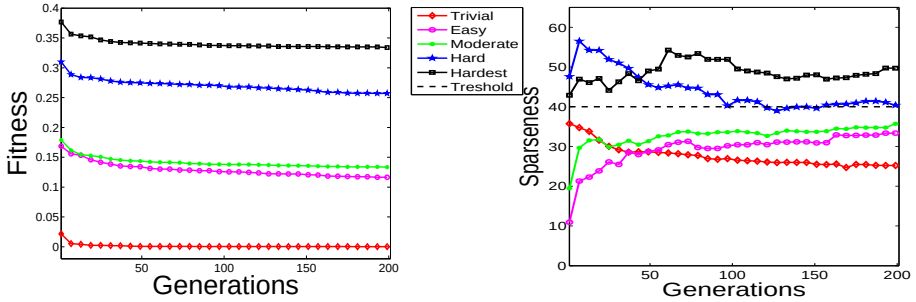
Fig. 7. Evolution of NS-CD with parameters $k = 15$ and $\rho_{min} = 40$.



(a) *Fitness* (b) *Sparseness*

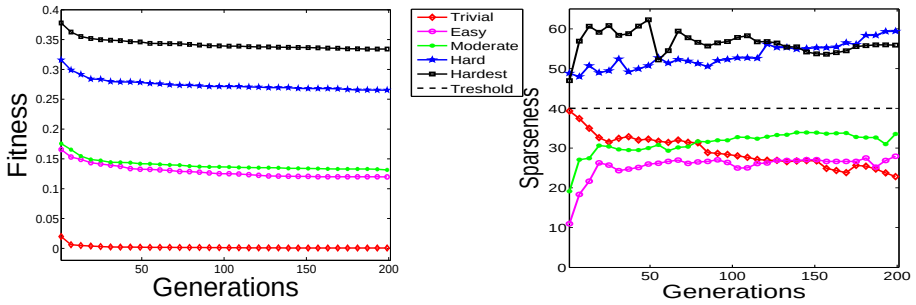
Fig. 8. Evolution of NS-CD-L with parameters $k = 15$ and $\rho_{min} = 40$.

6. 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.
7. J. Lehman and K. O. Stanley. Abandoning objectives: Evolution through the search for novelty alone. *Evol. Comput.*, 19(2):189–223, 2011.
8. 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.
9. J. B. Mouret and S. Doncieux. Encouraging behavioral diversity in evolutionary robotics: An empirical study. *Evol. Comput.*, 20(1):91–133, 2012.
10. C. Ofria and C. O. Wilke. Avida: a software platform for research in computational evolutionary biology. *Artif. Life*, 10(2):191–229, 2004.
11. 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.
12. 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.



(a) *Fitness* (b) *Sparseness*

Fig. 9. Evolution of NS-AD with parameters $K = 15$ and $S = 40$.



(a) *Fitness* (b) *Sparseness*

Fig. 10. Evolution of NS-AD-L with parameters $K = 15$ and $S = 40$.

13. 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.
14. 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.
15. 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.
16. B. G. Woolley and K. O. Stanley. Exploring promising stepping stones by combining novelty search with interactive evolution. *CoRR*, abs/1207.6682, 2012.
17. 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.