Generalization in Maze Navigation Using Grammatical Evolution and Novelty Search

Paulo Urbano[1,⋆], Enrique Naredo[2], and Leonardo Trujillo[2]

[1] LabMAg, Universidade de Lisboa, 1749-016 Lisbon, Portugal
pub@di.fc.ul.pt
[2] Tree-Lab, Instituto Tecnológico de Tijuana, Calz. del Tecnológico S/N, Tomás Aquino, 22414 Tijuana, Baja California, México
enriquenaredo@gmail.com, leonardo.trujillo@tectijuana.edu.mx

Abstract. Recent research on evolutionary algorithms has begun to focus on the issue of generalization. While most works emphasize the evolution of high quality solutions for particular problem instances, others are addressing the issue of evolving solutions that can generalize in different scenarios, which is also the focus of the present paper. In particular, this paper compares fitness-based search, Novelty Search (NS), and random search in a set of generalization oriented experiments in a maze navigation problem using Grammatical Evolution (GE), a variant of Genetic Programming. Experimental results suggest that NS outperforms the other search methods in terms of evolving general navigation behaviors that are able to cope with different initial conditions within a static deceptive maze.

Keywords: Novelty Search, Grammatical Evolution, Genetic Programming.

1 Introduction

Genetic Programming (GP) is a machine learning approach for the discovery of computer programs through an evolutionary search process. An important evaluation criteria for artificial learning systems, and for GP in particular, is their ability to find high quality solutions. However, generalization is also crucial, and several works have been devoted to this issue [2,7,6]. A general solution is one that is able to have a high performance on cases used for learning, and also on newer unseen cases. For example, in maze navigation, an artificial agent must find its way through the maze, from the start to a target point. An evolved control program may be successful in solving a particular navigation task, able to reach the target starting from a fixed point within the maze, but unable to solve the task when we change some aspect of the problem such as the initial conditions or environmental structure. In this scenario, the learned program is considered to be overfitted to the particular scenario used for learning. Ideally we would like to evolve programs that exhibit general navigation behaviors; i. e., performs well within a wide range of previously unseen maze scenarios.

⋆ Corresponding author.

A.-H. Dediu et al. (Eds.): TPNC 2014, LNCS 8890, pp. 35–46, 2014.

Some maze navigation tasks are used as deceptive benchmark problems for traditional Evolutionary Algorithms (EA). The wall configurations of mazes may create occlusions and *cul-de-sacs*, complicating the navigation task. Suppose that fitness is proportional to the Euclidean distance of a robot to the target. Then, when the walls of the mazes are obstacles that block the direct path to the target, the fitness gradient does not lead towards a feasible direction, thus deceiving the evolutionary search process, towards a local optima.

Therefore, to solve a deceptive maze navigation task, the search process must diverge from areas of high fitness and explore areas of low fitness. The problem with fitness-based EA is that, by following the gradient of the fitness function, it will not reward low fitness individuals, thus failing to reach the target. In EA literature, the main methods for avoiding local optima have focused on the promotion of genomic diversity [15,1]. Recently, rewarding diversity in the space of behaviors has received growing attention [12]. In general, all diversity-preserving algorithms take fitness into account, but Novelty Search (NS), a recent divergent evolutionary technique, takes a more unique step.

NS ignores the fitness function explicitly, relying only on behavioral diversity as the sole criteria for selection and reproduction. Therefore, instead of guiding the search for the *fittest* individual, NS guides the search for the most novel individual, replacing fitness measure by a novelty score taken from the individuals' behavior description. NS explores the behavioral space without any goal, besides generating novelty, ultimately an individual with the desired behavior may be found. NS has been successfully applied in several deceptive problems in neuro-evolution [5,10], and in GP [9,14,13] in some cases outperforming fitness-based EA.

However, most of the research on NS has ignored the issue of generalization. The motivation of the current paper is to study the capabilities of NS regarding generalization using maze navigation and Grammatical Evolution (GE) [16]. We use a fixed maze and a fixed target point for training, and we vary the starting point and orientation during testing. The goal is to test how general are the behaviors when they are evolved using only a single training instance, and also if different initial conditions have implications on the generalization of the solutions. We also used a bigger training set composed of several initial conditions, with the goal of evolving behaviors that are able to solve the maze task for every training case. The generalization abilities of the best evolved behaviors are also evaluated in the same test set. The performance of NS on the train and test set is compared against fitness based and random search. Our hypothesis is that an heterogeneous training set might originate a second level of deception to traditional fitness-based evolution. Some instances of the training set will be easier to solve than others and fitness-based evolution might tend to reward individuals fitted to the easiest training scenarios, failing to generalize. We also hypothesize that NS might play a significant role in the evolution of general maze navigation agents, preventing evolution to get trapped in local optima.

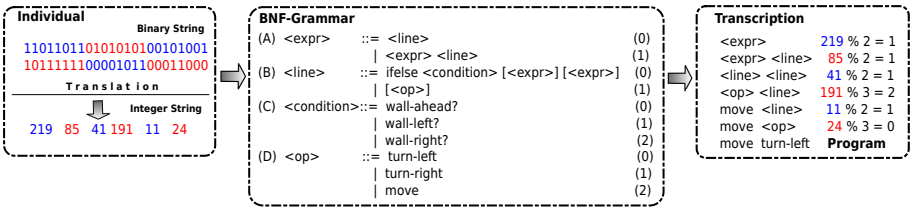

Fig. 1. Example of a GE genotype-phenotype mapping, where the binary genotype is translated into an integer string used to select production rules from a grammar. The derivation sequence of the program is shown on the right. All codons were used but wrapping was unnecessary.

2 Grammatical Evolution

Grammatical Evolution (GE) is an evolutionary algorithm that can evolve computer programs in an arbitrary language [16] defined by a Backus-Naur Form (BNF) grammar. Programs are indirectly represented by variable length binary genomes, and are built in a development process. The genome linear representation allows the application of genetic operators such as crossover and mutation in a typical genetic algorithm manner, unlike in the standard tree-based GP approach. Beginning with the start symbol of the grammar, a genotypephenotype mapping is employed such that each individual's binary genome, contains in its codons (typically groups of 8 bits) the information to select and apply the grammar production rules and generate a program.

Production rules for each non-terminal will be indexed starting from 0. In order to select a production rule for the left-most non-terminal of the developing program, from left to right the next codon value on the genome is read and interpreted using the following formula: $I = c\%r$, where c represents the current codon value, % represents the modulus operator, and r is the number of production rules for the left-most non-terminal. The correspondent production in the I-th index will be used to replace the left-most non-terminal. If, while reading codons, the algorithm reaches the end of the genome, a wrapping operator is invoked and it starts reading again from the beginning of the genome. The process stops when all of the non-terminal symbols have been replaced, resulting in a valid program. In the wrapping process, if an individual fails with this condition after a maximum of wraps, then it is considered an invalid individual, and is given the lowest score. The mapping process is illustrated with an example in Figure 1, where we use a Grammar for describing maze navigation programs written in Netlogo [19].

3 Novelty Search

Implementing Novelty Search [8] requires little change to any evolutionary algorithm aside from replacing the fitness function with a domain dependent novelty

metric. The novelty metric measures how different an individual is from other individuals with respect to behavior. In NS, there is a constant evolutionary pressure towards behavioral innovation. The behavior of each individual is normally characterized bey a vector of real numbers that capture behavior information along the whole evaluation or which is just sampled at some particular instants. The novelty of an individual is measured with respect to the behaviors of the current population and of a sample of predecessor individuals, stored in an archive. The archive is initially empty, and new behaviors are added to it if they are significantly different from the ones already there, i.e., if their novelty is above a dynamically computed threshold.

The novelty metric characterises how far the new individual is from the rest of the population and its predecessors in behavior space, based on the sparseness at the respective point in the behavior space. A simple measure of sparseness at a point is the average distance to the k-nearest neighbours at that point, where k is a constant empirically determined. Intuitively, if the average distance to a given point's nearest neighbours is large then it is in a sparse area; it is in a dense region if the average distance is small. The sparseness at each point is given by Equation 1, where μ_i is the ith-nearest neighbour of x with respect to the behavior distance metric $dist$, which typically is the Euclidean distance between domain-dependent behavior characterisation vectors.

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

Candidates from more sparse regions of this behavioral search space then receive higher novelty scores, thus guiding the search towards what is new, with no other explicit objective.

4 Related Work

With the exception of [2,11], as far as we know, the research on NS has mostly ignored the issue of generalization in robotic domains. For example, Lehman and Stanley in neuro-evolution [9], Lehman and Stanley in GP [10], as well as Loukas and Georgiou in GE [4], have made experiments using a variety of mazes with different levels of deception, but the evolved behaviors were specific for each maze configuration and initial conditions. It was not tested if the evolved behaviors were able to generalize to different starting and target points or to different mazes. Velez and Clune in [18] transfered maze navigation robots evolved with NS to new scenarios. Their experiments in neuro-evolution have confirmed that agents using NS do learn exploration skills. The transferred robots did in fact perform much better than randomly generated agents but did not outperform transferred robots evolved by a standard fitness-based EA. In [9], using standard GP, and in [17], using Grammatical Evolution, NS was applied successfully to the Santa Fe Trail Problem (SFT), a known deceptive problem in GP, but their goal was finding individuals able to perform well in that trail. Kushchu [7]

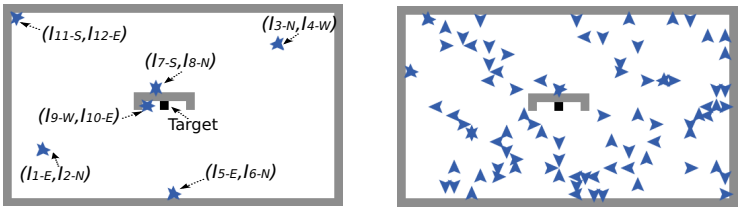

Fig. 2. The Medium Maze is a rectangle of 38 × 22 tiles. Left figure shows the target represented by a black square, and the 12 training instances labelled from I_1 to I_{12}. On each location there are 2 instances with different orientations. The initial orientations are labeled by: N=North, E=East, S=South, W=West. The right figure shows the 100 initial conditions used for testing, randomly generated.

has identified the SFT problem as an example of evolution of brittle solutions in fitness-based GP. His experimental results showed that most of the time, a successful ant won't perform well on some variations of the same trail. Kushchu in [7] proposed to train an ant on a set composed of variations of the SFT, sharing similar characteristics and tested the learned behaviors on a different set of similar trails. He was able to successfully evolve general trail following ant behaviors for a class of trails similar in difficulty to the SFT. Doucette and Heywood [2] have empirically evaluated the impact of NS on generalization performance for the SFT, using SFT as a single set and a test set of similar trails. They evaluated a cross-section of combined novelty and fitness, and fitness only function, and no method was to produce successful individuals in both the SFT and the trails from the set. However, results showed that the classical fitness-based GP provided best train and test performances, but programs evolved by NS alone had more generalization abilities, i.e., lower differences between train and test performance. In contrast, in two other experiments [9,17] NS outperformed fitness-search in the SFT.

5 Maze Navigation Experiments

Given the static maze shown in Figure 2, similar to the Medium Maze of [9], an agent controlled by a GE program must navigate from a specific starting point and orientation to a target point using a limit number of moves. The chosen maze has some potential for deception as the target is behind an inner wall blocking direct paths. The agent may sense the wall in the square in front, in the square on the right and on the left. It has 3 possible actions each one consuming one move: it may move forward one square if is there is no wall in front, it may turn right or left, rotating 90 degrees, clockwise or counter clockwise, respectively.

The grammar that defines the space of possible programs is given in Figure 1, the one we used to illustrate genotype to phenotype mapping in GE. We used 3 sensor boolean functions (wall-ahead?, wall-left? and wall-right?), and three actions (move, turn-left and turn-right). The program will be repeatedly

Table 1. Parameters used for both experimental setups. Codons-min and Codons-max are the minimal and maximal number of codons in the initial random population.

Parameter	Value	Parameter	Value
Codon-size	8	Generational	YES
Codons-min	15	Mutation prob	0.01
Codons-max	25	Elitism	10%
Number of runs	100	NS archive	NO
Wraps	10	Crossover	codon-crossover
Number of individuals	250	NS $k-$neighbors	3
Crossover prob.	0.9	Maximum of moves	$100,500$
		Selection	Roulette Wheel

executed until the agent hits the target or reaches the maximum number of moves. The agent succeeds if it hits the target square within the fixed limit of moves.

We compared novelty, fitness and random-based search in a series of generalization experiments. We want to assess how general the evolved behaviors are using only a single training instance. We have tried out different starting conditions, i.e., different single instance training sets. We have chosen 12 initial conditions for training the agent shown in the picture on the left side of Figure 2. Training instances were chosen to be heterogeneous in terms of the level of deception and difficulty imposed by the navigation task. Some starting points will be more or less blocked by walls, may be more or less distant to the target point, adjacent to the wall with the target or near the external wall, far from the target, others will be in the empty space, distant from both walls. For each initial condition, we have evaluated, in a series of runs, the performance of the best program from each run on an independent test set of initial conditions, not used for learning. For testing, we used the 100 initial positions and orientations presented on the right of Figure 2, that were randomly generated but correspond to a wide range of initial conditions. We also used a training set composed with all of the 12 initial conditions, and evaluate the generalization abilities in the same test set with 100 instances. Our objective is to evolve behaviors that are successful for all 12 initial conditions and evaluate their generalization abilities.

All experiments mentioned in this study are performed using the jGE library [3], which is a Java implementation of GE, and jGE Netlogo [4], which is a Netlogo extension of the jGE library. The Netlogo program was extended with an implementation of NS. The phenotype is a NetLogo program and the space of possible programs is given by the BNF-grammar showed on Figure 1. The experiments were repeated 100 times with a population of 250 individuals for 50 generations. The parameters used are presented in Table 1.

Fitness-based search needs a measure of performance to evaluate individuals, thus fitness is computed by $1/(1 + \text{dist}(p, q))$, where p is the final position, and q the maze target. In the case of a training data with 12 instances, each agent is evaluated 12 times and the final fitness is the average score across all the evaluations. Similarly, the fitness on the test set will be the average fitness across all 100 evaluations.

NS needs a behavior descriptor, so we used same descriptor from [10], which is the final position of the robot after hitting the target point or exhausting the maximal number of moves. By ignoring the details of the trajectories, NS will rewards agents that end in zones where nobody or less agents have ended before. In the case of a training set with 12 instances, the agent is evaluated 12 times, for every instance of the training set, and its final behavior descriptor is composed by the 12 ending points, which are concatenated in a vector to obtain a single descriptor. For an invalid individual, a value of 0 is given for both fitness and novelty. In the NS extension of GE, the novelty score of an individual will be the average behavior distance towards the behaviors of its k-nearest valid neighbours. After some preliminary exploration, we set to 3 the number of neighbours used to compute novelty score and we did not use an archive as in [8] since in our experiments it did not help to improve the performance.

6 Results

We begin by presenting and discussing results gathered from experiments where the agents are limited to 100 moves following [9].

6.1 Results and Analysis for a Maximal of 100 Moves

The results have been separated into training and testing performance. Considering the best programs from each run, we measured the percentage of hits and the average fitness in the train and test sets. Results are illustrated in Table 3 for each training set composed of a single instance comparing all three methods. Although NS did not obtain a 100% percentage of hits for every instance, it had the best performance, followed by fitness-based search, and finally random search. As random search has attained almost maximal performance with the following set of initial instances: $\{I_5, I_6, I_7, I_8, I_9, I_{10}\}$, we may conclude that they define easy navigation tasks. The best behaviors evolved by the three methods performed poorly on the test set: Not a single behavior was able to hit the target for every testing instance and the average fitness scores were low and very similar for all three methods. The results obtained with experiments with a training set composed with the 12 instances presented in Table 2, show that none of the methods were able to evolve behaviors that hit the target for all 12 training instances, exhibiting also a very poor performance on the test set. Our explanation about the poor training performance of both fitness-based search and NS on on the 12 instances experiments, is that a maximum number of 100 movements, following [9], imposes a heavy constraint inhibiting maze exploration and thwarting the evolution of general maze navigation behaviors. Hence, we have repeated the experiments fixing a new limit for the number of moves: 500, which increases *time* for maze exploration.

6.2 Results and Analysis for a Maximal of 500 Moves

The results regarding training sets with a single instance are presented in Table 3. These results are better for novelty and fitness-based search when compared with

Table 2. Table of results from the training set composed with the 12 instances. L-100 and L-500 stands for the limit of moves. NS-K3 stands for Novelty Search with the k-neighbour parameter set to 3. Fit stands for the average fitness score of the best program from each run, and Hits for the average number of best programs that hit the target.

		Training		**Testing**	
		Hits	**Fit**	**Hits**	**Fit**
L-100	NS-K3	0%	0.46	0%	0.14
	Fitness	0%	0.40	0%	0.13
	Random	0%	0.39	0%	0.12
L-500	NS-K3	61%	0.93	41%	0.89
	Fitness	22%	0.63	13%	0.47
	Random	1%	0.40	0%	0.16

experiments with the number of moves limited to 100. On the other hand, the differences for random search are not so relevant: It was still able to get a 100% of successful solutions for some of the initial conditions, and a very low hit percentage and average fitness for the most difficult cases, similar to the experiment with a limit of 100 moves. NS outperforms fitness-based and random search in terms of training performance, having a 100% of hits for every training instance, and a higher average fitness. Fitness-based against random search shows better results for some of the instances, while for the instances I_5, I_7 it shows lower performance than random search. However, instances $\{I_1, I_2, I_3, I_4, I_{11}, I_{12}\}$ introduce more difficulty for the methods tested. Those points, when used as single training sets will generate behaviors with higher generalization abilities than other instances. In contrast the easiest starting conditions resulted in behaviors with the lowest generalization abilities. All three methods were still unable to evolve general behaviors after being trained with a single instance. Nevertheless, NS showed the best performance against fitness and random-based search, and the best test performance was 13% for I_{12}.

Results obtained from experiments with a training set of 12 instances are presented in Table 2. In this experiment, NS had the best performance again in terms of training and testing, 61% of the runs generate a program which hit the target from every initial condition in the training set, while 41% of the runs perform successfully in the test set. Fitness-based search had 22% of hits in the training set, and 13 exhibit general navigation skills which are able to cope with every initial condition in the test set. In terms of average fitness scores, fitness-based search, was clearly outperformed by NS: a difference of 0.3 in the training set and 0.43 in the test set. In a training set composed of instances that correspond to tasks with different levels of difficulty and deception, it will be easy to find individuals that solve the less difficult cases and it will harder to solve instances defining more deceptive tasks, creating local optima for fitness-based search.

Furthermore, one interesting observation regarding both limits of moves: $L - 100$ and $L - 500$, is that the orientations in some of the initial conditions are

```
move move turn-left
ifelse  wall-left?
 [tu rn-left]
 [move move ifelse wall-ahead?
   [ turn-right  move move ]
   [ move  turn-right]  ]
```

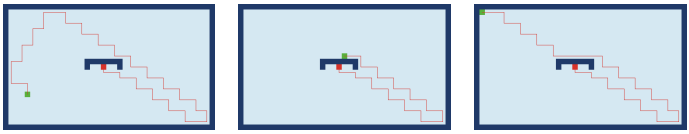

Fig. 3. Left figure shows an example of an evolved program able to solve the maze for every test instance, exhibiting general navigation. Figures at the right show the trajectory when using this program for initial conditions: I_1, I_7, I_{11}, respectively.

Table 3. Table of results from each of the twelve instances, where the sub-index stands for the number of the training instance. L-100, and L-500 stands for the limit of moves. NS-K3 stands for Novelty Search with the k-neighbour parameter set to 3. Fit stands for the average fitness score, and Hits for the percent of best individuals which reach the target.

		L-100				**L-500**			
Instance		**Train**		**Test**		**Train**		**Test**	
	Method	Hits	Fit	Hits	Fit	Hits	Fit	Hits	Fit
I_1	NS-K3	100%	1.000	0%	0.110	100%	1.00	3%	0.263
	Fitness	94%	0.950	0%	0.112	94%	0.74	3%	0.286
	Random	19%	0.330	0%	0.111	34%	0.44	1%	0.149
I_2	NS-K3	100%	1.000	0%	0.133	100%	1.00	6%	0.365
	Fitness	73%	0.796	0%	0.126	69%	0.95	2%	0.201
	Random	10%	0.233	0%	0.102	14%	0.28	2%	0.159
I_3	NS-K3	77%	0.864	0%	0.166	100%	1.00	12%	0.476
	Fitness	7%	0.295	0%	0.166	62%	0.69	1%	0.328
	Random	0%	0.129	0%	0.115	8%	0.20	0%	0.145
I_4	NS-K3	92%	0.949	0%	0.141	100%	1.00	8%	0.455
	Fitness	22%	0.430	0%	0.124	68%	1.00	3%	0.665
	Random	0%	0.149	0%	0.124	10%	0.22	1%	0.165
I_5	NS-K3	100%	1.000	0%	0.079	100%	1.00	0%	0.091
	Fitness	96%	0.980	0%	0.073	95%	0.98	0%	0.072
	Random	100%	1.000	0%	0.074	100%	1.00	0%	0.083
I_6	NS-K3	100%	1.000	0%	0.010	100%	1.00	4%	0.168
	Fitness	86%	0.930	0%	0.081	90%	0.95	2%	0.107
	Random	83%	0.915	0%	0.089	86%	0.93	0%	0.106
I_7	NS-K3	100%	1.000	0%	0.120	100%	1.00	0%	0.124
	Fitness	100%	1.000	0%	0.117	99%	0.99	0%	0.125
	Random	100%	1.000	0%	0.118	100%	1.00	0%	0.120
I_8	NS-K3	100%	1.000	0%	0.114	100%	1.00	0%	0.121
	Fitness	100%	1.000	0%	0.113	100%	1.00	0%	0.120
	Random	100%	1.000	0%	0.113	100%	1.00	0%	0.115
I_9	NS-K3	100%	1.000	0%	0.098	100%	1.00	0%	0.095
	Fitness	100%	1.000	0%	0.099	100%	1.00	0%	0.096
	Random	100%	1.000	0%	0.097	100%	1.00	0%	0.095
I_{10}	NS-K3	100%	1.000	0%	0.082	100%	1.00	0%	0.081
	Fitness	100%	1.000	0%	0.083	100%	1.00	0%	0.081
	Random	100%	1.000	0%	0.082	100%	1.00	0%	0.084
I_{11}	NS-K3	60%	0.741	0%	0.160	100%	1.00	11%	0.538
	Fitness	8%	0.264	0%	0.128	57%	0.64	6%	0.387
	Random	0%	0.139	0%	0.111	7%	0.19	2%	0.173
I_{12}	NS-K3	68%	0.793	0%	0.163	100%	1.00	13%	0.627
	Fitness	7%	0.274	0%	0.116	59%	0.67	2%	0.422
	Random	1%	0.140	0%	0.114	6%	0.19	2%	0.174

in fact relevant in the training set as well as in the test set, since they impact on the overall generalization performance. We can see in Table 3 differences in train and test performance between some pairs of instances: I1 and I2, I3 and I4.

Figure 3 shows an example of one of the best evolved solutions, and the trajectories for 3 different initial conditions.

7 Conclusions and Future Work

This work presents the first application of NS with GE to study their generalization abilities on a maze navigation task. An agent controlled by a GE program must navigate in a maze from a specific starting point and orientation to a target point using a limit number of moves. We have compared novelty, fitness and random based evolution using GE in a series of generalization experiments. We have used a fixed maze a fixed target and varied the agent initial position and orientation. The goal is to evolve behaviors that are able to hit the target starting from any point and facing any direction. In this work, we consider 12 different initial conditions for the agent, where some positions are easier or less deceptive than others. First, we start by using just one instance at a time, then we use all 12 instances as training set, and the evolved programs were tested on a set of 100 random instances. Furthermore, we use two different moves limit: 100 and 500. The experiments with a limitation of 100 moves showed that it was not possible to evolve programs with general navigational abilities. The performances exhibited by the best programs evolved by the three methods were very similar and very poor in the test set. The three methods were also unable to evolve a single program able to hit the target for every starting condition in the training set composed with the 12 instances. Anyhow, NS exhibited the best performance, followed by fitness based search. Regarding evolution using single instances training sets, NS outperformed fitness based search and also random search for the more deceptive cases, as it was expected, since it does not follow the gradient of the fitness function, but all showed similar results in the easier cases. Anyhow, all successful behaviors for single training instances were too overfitted and failed in the test set.

When we increase the limit of moves to 500, allowing more time for exploration, every method is still unable to evolve general navigation abilities for the single instance training sets, composed with the easier starting conditions. In contrast, all three methods were able to evolve programs with general navigation abilities, able to solve the maze task for every condition in the test set, using single instance training sets that impose more difficulty and deception. This happened not so frequently but more frequently with NS than with the other two methods. Therefore, instances that appear to impose a higher difficulty in the navigation task, when used alone in the training set, seem to induce better general navigation skills using all three methods. But, if we use a more numerous training set, using the 12 training instances all together, random based search was unable to find general behaviors that solve the maze task for every instance of the training set and the same happened in the test set. NS had the best performance both in the training set (12 instances) and in the test set evolving more frequently general navigation programs. Fitness based search seems to be overfitting to some of the instances of the training set, the easiest ones, which is enough to achieve a higher score, creating a local optima.

NS in these experiments exhibits a substantial improvement in the evolution of general maze navigation agents, avoiding the deception involved in the maze task, preventing evolution from being trapped in local optima. This research work presents a window of opportunity for generalization research, for instance, instead of having fixed targets we may perform generalization experiments where we vary both the initial conditions and the target of maze tasks. Additionally, we may try to transfer agents evolved with NS to new mazes to test if the skills acquired in a particular environment generalize to unseen environments. Finally, in a general way, one further work will be trying to understand what must be the right ingredients to get the right training set in order to evolve successfully general behaviors.

Acknowledgments. The authors acknowledge the following projects. First author is supported by FCT project EXPL/EEI-SII/1861/2013. Second author is supported by CONACYT (México) scholarship No. 232288. Third author is supported by CONACYT (México) Basic Science Research Project No. 178323, DGEST (México) Research Projects No.5149.13-P, also by TIJ-ING-2012-110, and by FP7-Marie Curie-IRSES 2013 project ACoBSEC funded by the European Comission.

References

1. Burke, E.K., Gustafson, S., Kendall, G., Krasnogor, N.: Is Increased Diversity in Genetic Programming Beneficial? An Analysis of Lineage Selection. Ph.D. thesis, University of Nottingham, UK (February 2004)
2. Doucette, J., Heywood, M.: Novelty-based fitness: An evaluation under the santa fe trail. Genetic Programming, 50–61 (2010)
3. Georgiou, L., Teahan, W.J.: jge - a java implementation of grammatical evolution. In: 10th WSEAS International Conference on Systems, Athens, Greece, pp. 534–869 (2006)
4. Georgiou, L., Teahan, W.J.: Grammatical evolution and the santa fe trail problem. In: International Conference on Evolutionary Computation (ICEC), pp. 10–19. SciTePress, Valencia (2010)
5. Gomes, J.C., Urbano, P., Christensen, A.L.: Evolution of swarm robotics systems with novelty search. CoRR abs/1304.3362 (2013)
6. Gonçalves, I., Silva, S.: Experiments on controlling overfitting in genetic programming. In: 15th Portuguese Conference on Artificial Intelligence (EPIA 2011) (2011)
7. Kushchu, I.: Genetic programming and evolutionary generalization. IEEE Transactions on Evolutionary Computation 6(5), 431–442 (2002)
8. Lehman, J., Stanley, K.: Exploiting open-endedness to solve problems through the search for novelty. In: Bullock, S., Noble, J., Watson, R., Bedau, M.A. (eds.) Artificial Life XI: Proceedings of the Eleventh International Conference on the Simulation and Synthesis of Living Systems, pp. 329–336. MIT Press, Cambridge (2008)
9. Lehman, J., Stanley, K.O.: Efficiently evolving programs through the search for novelty. In: Pelikan, M., Branke, J. (eds.) GECCO, pp. 837–844. ACM (2010)

10. Lehman, J., Stanley, K.O.: Abandoning objectives: Evolution through the search for novelty alone. Evolutionary Computation 19(2), 189–223 (2011)
11. Li, J., Storie, J., Clune, J.: Encouraging creative thinking in robots improves their ability to solve challenging problems. In: Proceedings of the 2014 Conference on Genetic and Evolutionary Computation, GECCO 2014, pp. 193–200. ACM (2014)
12. Mouret, J.B., Doncieux, S.: Encouraging behavioral diversity in evolutionary robotics: An empirical study. Evolutionary Computation 20(1), 91–133 (2012)
13. Naredo, E., Trujillo, L.: Searching for novel clustering programs. In: GECCO, pp. 1093–1100 (2013)
14. Naredo, E., Trujillo, L., Martínez, Y.: Searching for novel classifiers. In: Krawiec, K., Moraglio, A., Hu, T., Etaner-Uyar, A.Ş., Hu, B. (eds.) EuroGP 2013. LNCS, vol. 7831, pp. 145–156. Springer, Heidelberg (2013)
15. Nicoară, E.S.: Mechanisms to avoid the premature convergence of genetic algorithms. Petroleum - Gas University of Ploiesti Bulletin, Mathematics LXI(1) (2009)
16. O'Neill, M., Ryan, C.: Grammatical evolution. IEEE Trans. Evolutionary Computation 5(4), 349–358 (2001)
17. Urbano, P., Loukas, G.: Improving grammatical evolution in santa fe trail using novelty search. In: Advances in Artificial Life, ECAL, pp. 917–924 (2013)
18. Velez, R., Clune, J.: Novelty search creates robots with general skills for exploration. In: Proceedings of the 2014 Conference on Genetic and Evolutionary Computation, GECCO 2014, pp. 737–744. ACM (2014)
19. Wilensky, U.: Netlogo, Evanston, IL: Center for Connected Learning and Computer-Based Modeling (1999), `http://ccl.northwestern.edu/netlogo`