DOI: http://dx.doi.org/10.7551/978-0-262-31709-2-ch137

Improving Grammatical Evolution in Santa Fe Trail using Novelty Search

Paulo Urbano[1] and Loukas Georgiou[2]

[1]LabMAg - FCUL, Lisboa Portugal
pub@di.fc.ul.pt
[2]luc_georgiou@hotmail.com

Abstract

Grammatical Evolution is an evolutionary algorithm that can evolve complete programs using a Backus Naur form grammar as a plug-in component to describe the output language. An important issue of Grammatical Evolution, and evolutionary computation in general, is the difficulty in dealing with deceptive problems and avoid premature convergence to local optima. Novelty search is a recent technique, which does not use the standard fitness function of evolutionary algorithms but follows the gradient of behavioral diversity. It has been successfully used for solving deceptive problems mainly in neuro-evolutionary robotics where it was originated. This work presents the first application of Novelty Search in Grammatical Evolution (as the search component of the later) and benchmarks this novel approach in a well-known deceptive problem, the Santa Fe Trail. For the experiments, two grammars are used: one that defines a search space semantically equivalent to the original Santa Fe Trail problem as defined by Koza and a second one which were widely used in the Grammatical Evolution literature, but which defines a biased search space. The application of novelty search requires to characterize behavior, using behavior descriptors and compare descriptions using behavior similarity metrics. The conducted experiments compare the performance of standard Grammatical Evolution and its Novelty Search variation using four intuitive behavior descriptors. The experimental results demonstrate that Grammatical Evolution with Novelty Search outperforms the traditional fitness based Grammatical Evolution algorithm in the Santa Fe Trail problem demonstrating a higher success rates and better solutions in terms of the required steps.

Background

Grammatical Evolution

Grammatical Evolution (O'Neill and Ryan, 2001) is an evolutionary algorithm that can evolve complete programs in an arbitrary language using populations of variable-length binary strings. Namely, a chosen evolutionary algorithm (typically a variable-length genetic algorithm) creates and evolves a population of individuals and the binary string (genome) of each individual determines which production rules in a Backus Naur Form (BNF) grammar definition are used in a genotype-to-phenotype mapping process to generate a program. In natural biology, there is no direct mapping between the genetic code and its physical expression. Instead, genes guide the creation of proteins, which affect the physical traits either independently or in conjunction with other proteins (Ryan et al., 1998). Grammatical Evolution treats each genotype to phenotype transition as a "protein" which cannot generate a physical trait on its own. Instead, each one protein can result in a different physical trait depending on its position in the genotype and consequently, the previous proteins that have been generated.

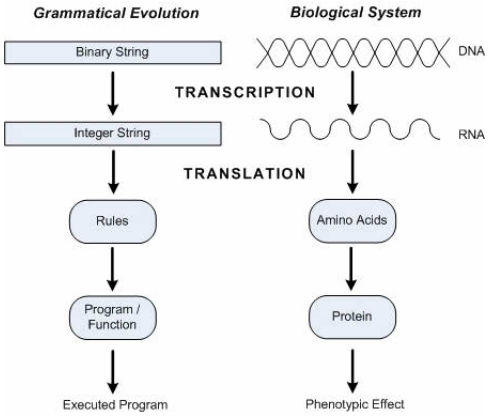

Figure 1: Comparison between the GE system and a biological genetic system. Cited in (O'Neill and Ryan, 2001), p.351.

Fig. 1 shows the comparison between the GE system and a biological genetic system (O'Neill and Ryan, 2001). The binary string of the genotype of an individual in GE is equivalent to the double helix of DNA of a living organism, each guiding the formation of the phenotype. In the case of GE, this occurs via the application of production rules to generate the terminals of the resulted program (phenotype) and in the biological case, directing the formation of the phenotypic protein by determining the order and type of protein subcomponents (amino acids) that are joined together.

Before the evaluation of each individual, the following steps take place in Grammatical Evolution:

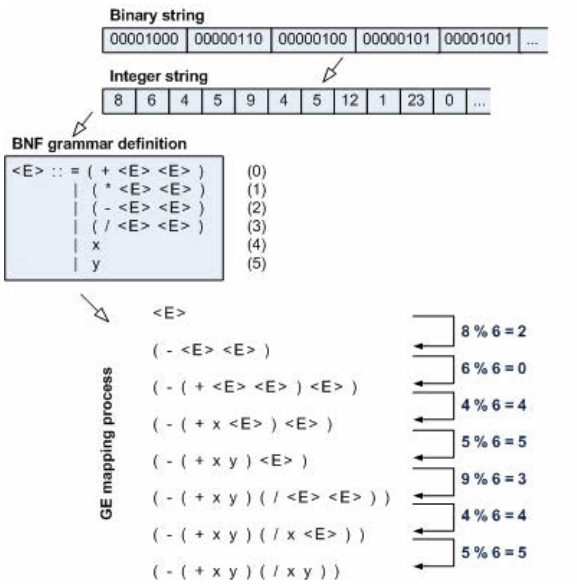

Figure 2: GE mapping process. From Dempsey et al. (2006), p. 2588, (with minor changes)

i The genotype (a variable-length binary string) is used to map the start symbol of the BNF grammar definition into terminals. The grammar is used to specify the legal phenotypes.

ii The GE algorithm reads "codons" of typically 8 bits (integer codons are also widely adopted) and the integer corresponding to the codon bit sequence is used to determine which form of a rule is chosen each time a non-terminal to be translated has alternative forms. If while reading "codons", the algorithm reaches the end of the genotype, it starts reading again from the beginning of the genotype (wrapping).

iii The form of the production rule is calculated using the formula *form = codon mod forms* where *codon* is the codon integer value, and *forms* is the number of alternative forms for the current non-terminal.

An example of the mapping process employed by Grammatical Evolution is shown in Fig. 2. In this example, the first codon of the genotype of the individual is the binary string *00001000* which is the binary form of the integer 8. The start symbol *<E>* has six alternative forms, therefore the form to be applied is this with label 2 (8 % 6) which results in the expression *(- <E> <E>)*. The next codon is then read in order to replace the first non-terminal symbol *<E>* of the new expression, and this goes on until the expression contains only the terminal symbols x and y and any of the arithmetic operators. Namely, until all of the non-terminal symbols have been replaced.

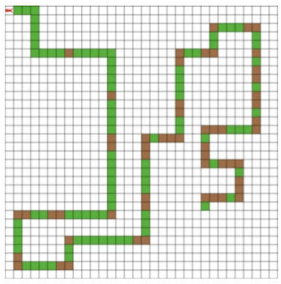

Figure 3: "The Santa Fe Trail.

After the mapping process (i.e. the creation of the phenotype), the fitness score is calculated and assigned to each individual (phenotype) according to the given problem specification. These fitness scores are sent back to the evolutionary algorithm which uses them to evolve a new population of individuals. Grammatical Evolution is a flexible and promising grammar-based evolutionary algorithm with two unique features inspired from molecular biology: genetic code degeneracy which improves genetic diversity, due to its many-to-one genotype-to-phenotype mapping mechanism; and genetic material reuse, due to the genome wrapping it applies. Even though Grammatical Evolution shows competence on a series of problems where it has been applied, the experiments conducted by Georgiou and Teahan (2010, 2011) and Georgiou (2012) cast doubt about its effectiveness and efficiency on deceptive problems such as Artificial Ant and Maze searching.

The Santa Fe Trail Problem

The Santa Fe Trail is the most common instance of the Artificial Ant problem and was designed by Christopher Langton (Koza, 1991). It is a standard and challenging problem, which is widely used for benchmarking in Genetic Programming (Koza, 1991, 1992) and Grammatical Evolution (O'Neill and Ryan, 2001, 2003).

This problem can be briefly described as finding a computer program to control an artificial ant, within a limited amount of steps, such that it can find all food items forming a twisting trail in a 32 x 32 toroidal plane grid. The trail is composed by 89 food pellets distributed non-uniformly along it, and has 55 gaps and 21 turns (see Fig. 3). The Santa Fe Trail has the following irregularities: single gaps, double gaps, single gaps at corners, double gaps at corners (short knight moves) and triple gaps at corners (long knight moves) (Koza, 1991).

The artificial ant starts in the upper left cell of the grid (0,0), facing east, and can perform three primitive actions: *move, turn right* and *turn left* - each action consumes one step. The *move* action moves the ant forward one square in the direction it is currently facing. When the ant moves into a square, it eats the food if there is any. The other two actions

turn the ant right or left respectively by 90 degrees, without moving the ant. The ant can use a binary sensing boolean operator *food ahead*, which comes with no cost in moves. This sensing operator looks into the square the ant is currently facing and executes one of its arguments depending upon whether the square ahead contains food or is empty. Besides the sensing operator, Koza (1991) introduced the unlimited sequence operator *progn* that executes its arguments in order. Alternatively, Koza (1992) introduced two and limited sequence *progn2* and *progn3*, taking two and three arguments respectively. This difference regarding the sequence operators used introduces a representational bias, but does not affect the ant control possibilities, since any progn subtree can be translated into semantically equivalent subtrees using *progn2* and *progn3* operators and vice versa (Robilliard et al., 2005).

During the evaluation of the ant, its program is iterated until exhausting the steps limit or eating all food items. There is no unanimity in the literature regarding the fixation of the maximal number of steps an ant is allowed to perform. Robilliard et al. (2005) note that Koza said he arbitrarily fixed to 400, but Langdon and Poli (1998) set the limit to 600 steps, assuming a possible mistype in Koza's paper. O'Neill and Ryan (2003) set the maximum number of steps to 615 and Georgiou (2012) benchmarks a variety of GE configurations using 650 steps.

As the goal is to collect the maximal number of food items laid down in the trail, the customary fitness function is the amount of food collected by the ant, or in the case of minimization, the number of pellets missed out of the total 89 on the trail.

This benchmark problem is still repeatedly used (Georgiou and Teahan, 2010; Lehman and Stanley, 2010; Doucette and Heywood, 2010) because of its interesting characteristics: it has a large search space, full of global and local optima and many plateaus riven with deep valleys, which may be indicative of real problem spaces (Langdon and Poli, 1998). More, there are low and middle order schemas which are required as stepping stones to build solutions but which are below average fitness. Langdon and Poli (1998) have shown that GP does not perform significantly better than random search on this problem since it contains multiple levels of deception, where solutions have low fitness neighbors. The search space is highly deceptive because it has many high scoring non-optimal programs that do not correspond to trail following behavior: random policies may collect many food items.

Performance of GE on the Santa Fe Trail Problem

O'Neill and Ryan (2003) compared Grammatical Evolution with Genetic Programming, using the Santa Fe Trail problem as a benchmark, fixing the steps upper limit to 615. They have used the grammar in Fig. 4 (from now on called BNF-O'Neill grammar) and the experiments results were favorable to GE.

```
<code>     ::= <line> | <code> <line>
<line>     ::= <condition> | <op>
<condition> ::= ifelse food-ahead
                [ <line> ][ <line> ]
<op>       ::= turn-left | turn-right | move
```

Figure 4: "BNF-ONeill grammar definition.

```
<expr> ::= <line> | <expr> <line>
<line> ::= ifelse food-ahead
           [ <expr> ][ <expr> ] | <op>
<op>   ::= turn-left | turn-right | move
```

Figure 5: "BNF-Koza grammar definition.

The comparison SFT experiments were later questioned by Robilliard et al. (2005). They have found that the search space of programs possible by BNF-O'Neill grammar was not semantically equivalent to the set of programs possible within the original SFT definition. The former is narrower: any original SFT program can be translated into a semantically equivalent BNF-O'Neill program but the converse is not true. Robilliard et al. (2005) argue that they found no solution in the Santa Fe Trail problem using Grammatical Evolution, up to and including 600 time steps in their experiments, and they note that almost all Grammatical Evolution publications mention an upper limit of 615 time steps. Only in O'Neill and Ryan (2001) with Grammatical Evolution is the limit reported as 600 steps, which Robilliard et al. (2005) claim is a mistype.

Georgiou and Teahan (2010) have made a series of experiments where GE gives very poor results in the Santa Fe Trail problem with a search space semantically equivalent to the search space used in the original problem (Koza, 1991). They have used the SFT-BAP grammar cited in Robilliard et al. (2005), from now on named BNF-Koza (see Fig. 5), which defines a search space of programs semantically equivalent to the Koza's original (Robilliard et al., 2005). Indeed, the same work proved experimentally that GE literature (O'Neill and Ryan, 2001, 2003) uses a BNF grammar which narrows the original search space and consequently gives an unfair advantage to GE when it is compared against GP in the SFT problem. As noted by Robilliard et al. (2005) the BNF-O'Neill grammar does not allow multiple <op> statements or sequences of <op> and <condition> statements in the branches of the <condition> production rule, in contrast with the first production rule of <line> in the BNF-Koza.

In the experiments of Georgiou and Teahan (2010) the maximum allowable steps for the ant was set to 650. The reason for this increase was to give Grammatical Evolution the chance to find more solutions using the investigated BNF grammars in order that the sample of the solutions found

and the comparison of the effects of these grammars on the performance of Grammatical Evolution being more encompassing. Furthermore, they have proved experimentally that during an evolutionary run GE is not capable of finding solutions using the BNF-O'Neill grammar definition with less than 607 steps. BNF-O'Neill defines a smaller search space, biasing the search towards areas where a solution may be found more easily (higher success rate) but with the cost of excluding areas where more efficient programs (using less steps) exist. Note that the easy incorporation of domain knowledge, by changing the grammar, and biasing the search, is a great advantage of the grammar representation used by GE. But we should be careful not to exclude good solutions (the ones which require less steps), as it is the case of BNF-O'Neill.

The best solution we know so far for the SFT problem, in GP or GE, requires only 337 steps, and it was generated by CGE (Georgiou and Teahan, 2011), one of the GE variations developed over the last years.

Promoting diversity in GP to overcome deception

Diversity promotion and maintenance, relying generally on the genotype and fitness function, has been the main approach to mitigate deception and stagnation in GP. A variety of different techniques have been proposed in order to preserve genotypic diversity, that make use of different measures to compare genotypes. We are going to refer only some of them: higher mutation rates (Banzhaf et al., 1996), larger population sizes (Ciesielski and Mawhinney, 2002), fitness shaping (Luke, 1998), aiding explicit objectives to promote genotypic diversity, together with a goal-oriented one, in a multi-objective scheme (de Jong et al., 2001), replacing the most similar programs (Ciesielski and Mawhinney, 2002), maintaining a diversity of genetic lineages (Burke et al., 2003) and fitness sharing (Ekárt and Németh, 2002). Burke et al. (2002) have shown that genotype diversity approaches manifest a low correlation with fitness.

A different approach is the promotion of phenotypic diversity based on fitness: the use of a selection method which is uniform over the fitness values (Legg and Hutter, 2005) or the phenotype diversity measurement in terms of the number of unique fitness values in the population, using entropy (Rosca, 1995). Burke et al. (2002) concluded that "successful evolution occurs when population converges to similar structure and high fitness diversity" and "the fitness based measures of phenotypes and entropy appear to correlate better with run performance." The phenotype techniques spread individuals across different fitness levels but are not able to maintain diversity in the same fitness levels (Legg and Hutter, 2005). Yan and Clack (2009) pointed out the importance of preserving phenotypic behavioral diversity, in achieving higher fitness and adaptability to dynamic domains, where behavior is not reduced to fitness, being much more detailed.

Novelty Search

All diversity preserving techniques, described above, take objective-based fitness into account. The Novelty Search (NS) approach, introduced by Lehman and Stanley (2008) is much more radical, as it ignores completely the fitness objective, relying only on behavior diversity as the sole criteria for selection in artificial evolution. Objective based fitness is replaced by novelty and the idea is not to select the most fitted individuals for reproduction but those with the most novel behaviors instead. Novel individuals are rewarded and will guide the search towards finding other novel individuals. NS is driven towards behavior space exploration, looking for what is divergent from the past and present behaviors, regardless of their fitness. The idea is that by exploring the behavior space without any goal besides novelty, ultimately an individual with the desired behavior will be found. Surprisingly, NS has been successful in several deceptive problems (Lehman and Stanley, 2011; Gomes et al., 2012), as it is not dependent of any fixed goal, avoiding the convergence towards local optima.

NS requires the definition of distances between behavior descriptors. Those descriptors may be specific to a task or suited for a class of tasks (Doncieux and Mouret, 2010). The descriptors are normally vectors that capture behavior information along the whole evaluation or which is just sampled at some particular instants. The used descriptors may even change along an evolutionary run. Several general distance functions between behavioral vectors have been suggested: euclidian distance, edit distance, hamming distance, relative entropy and normalized compression distance, fourier analysis, for instance (Doncieux and Mouret, 2010).

Given a behavior function and a distance metric, the novelty score of an individual is computed as the average distance from its k-nearest neighbors (μ_i) in both the population and the archive (see Eq. 1).

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

A point in a sparse area will be highly rewarded and a point in a dense area will receive a low novelty score. There have been several proposed ways of registering the past behaviors but we won't deal with them here, as in our NS application we wont use any archive of past behaviors.

Novelty Search applied to GP

Lehman and Stanley (2010) were the first to apply novelty search to Genetic Programming. They made experiments in three deceptive tasks: maze navigation, and the two artificial ant benchmarks: the Santa Fe Trail and Los Altos. There is no reference to the features of the best individual evolved, in terms of the number of steps, genotype or phenotype length, but in the end, NS was able to avoid bloat, evolving smaller

programs and outperforming objective based fitness in terms of the number of successfully evolutionary runs.

Doucette and Heywood (2010) used SFT to study the effects of NS on performance and also on solution generalization. NS evolved programs, which eat less food items on average than the traditional fitness based method but they achieved better performance in terms of generalization to new trails.

Naredo et al. (2013) applied NS to evolve GP classifiers and obtained encouraging results when compared to canonical GP. NS exhibited the best results when confronted with difficult problems, but for simple problems, however, the explorative capacity of NS seemed to be a detriment to the search.

Santa Fe Trail Experiments

All experiments mentioned in this study have been performed using the jGE library (Georgiou and Teahan, 2006), which is a Java implementation of the Grammatical Evolution algorithm, and jGE Netlogo extension (Georgiou and Teahan, 2010), which is a Netlogo extension of the jGE library. We have used the customary fitness function: the number of food items eaten. An invalid individual has a fitness of 0.

In order to evaluate the performance of Novelty Search in the Santa Fe Trail benchmark, we have used both grammars, BNF-ONeill and BNF-Koza. The latter is used because it is semantically equivalent to the original formulation of SFT, and we have used BNF-ONeill also because it has been extensively used in GE literature, although it narrows the original search space excluding good solutions. In order to sample a larger number of solutions we have fixed the maximum allowable steps for the ant to 650, as in (Georgiou and Teahan, 2010).

Note that when Grammatical Evolution is being performed with Novelty Search (NS-GE) every individual must be evaluated to know its fitness score. But that score is not being used for individual selection. The fitness score is only used to evaluate NS-GE performance.

We did not add any archive, thus novelty was only tested against the other individuals in the current population. Our preliminary experiments showed that the addition of an archive of past behaviors did not improve the performance of NS without memory. Moreover, the archive would introduce a memory mechanism that does not exist in the standard GE. In order to track novelty GE requires a small change: the fitness function is replaced by the novelty metric.

Behavior descriptors

We have evaluated NS using several behavior descriptors, and all of them were compared against the standard GE for both grammars. There is no natural behavior descriptor: we have many possibilities and we wanted to make empirical comparisons between several of them. The behavior descriptors should ideally be compact in order to capture relevant behavior variation and condense the irrelevant ones. We have to be careful not to conflate the stepping-stones that will lead the evolution towards finding successful solutions We are going to present our four behavior descriptors (two of them are different samplings of the same behavior characteristic):

Amount of food eaten This is the behavior descriptor used by Lehman and Stanley (2010): the amount of food eaten is sampled N evenly spaced times during an individual evaluation. We have considered two values for N, in order to cover descriptors with different granularity and implying behavior spaces with different sizes: N=1 and N=26. In the former case, the amount of food is sampled only once: at the end of the 650 steps or when the food trail is depleted. A sample of 1 means a behavior space of size 90 and a lot of conflation: many different forms of eating the same amount of food will be considered identical. A sampling of 26 means that the amount of food will be registered every 25 steps. The search space of behaviors will be much larger but there are constraints: it will be impossible to eat more than 25 items in each period. An invalid individual or a valid individual that does nor eat any food after 650 steps will have a vector filled with 26 zeros or 1 zero depending on the sampling size. We have used the Euclidean distance for measuring similarity between behaviors.

Food eaten sequence This is a behavior descriptor similar to the one used by Doucette and Heywood (2010) in his SFT experiments. The behavior descriptor is a vector of size 89 x 2, containing the food item coordinates in the order in which they were eaten. The x and y coordinates of the first food item eaten occupy the first and second vector position, respectively. The 3rd and 4th vector positions are occupied respectively by the x and y coordinates of the second food item eaten. If the ant ate N food items, being N less than 89, the first N x 2 vector cells will be occupied by the N food item position coordinates and all the other vector cells will be filled by the x and y coordinates of the Nth food item position. We have to describe the behavior of ants that do not eat any food item and also the invalid individuals. Both will have a vector filled with 0s. Note that not all of the food position combinations are possible because there are natural constraints imposed on the space of behaviors by the number of maximal steps. For example, it will be impossible to have the food items in reverse order: going to the end of the trail and then following it upside down. We will use the Euclidian distance for measuring similarity between behaviors.

Steps sequence We have a vector of 89 cells where the nth cell will be filled by the number of steps necessary to eat the nth food item. In case the ant has eaten only N (<89) food items, every cell after the Nth will be filled by the dummy

value. For example, this vector [1,2,3,*,*,*] means that one food item was eaten after the first move, a second food item after the second move, a third food item after three steps and then no more food items were eaten. An invalid individual will have a vector filled with the value *. Note that again the behavior descriptor imposes constraints on the space of possible behaviors. The vector is always sorted in increasing order, except the dummy *. We have used a variation on the Euclidean distance (see Eq. 2) for measuring the distance between behavior vectors using this descriptor. The subtraction in the Euclidean formula is substituted by a function $dif(v, w)$ in order to deal with the dummy value * (absence of food). Consider two vectors v and w, each one with 89 cells: $v = (v_1, v_2, ..., v_{89})$ and $w = (w_1, w_2, ...w_{89})$.

$$dist(v, w) = \sqrt{\sum_{i=0}^{89} dif(v_i, w_i)^2} \quad (2)$$

We want to penalize, as being different, the behavior vectors that have less food items (more *s). This way the difference between a dummy value and any number of steps is 500. The difference function between the values from two corresponding cells is given by Eq. 3.

$$dif(x, y) = \begin{cases} 0 & \text{if x=* and y=*} \\ x - y & \text{if x}\neq\text{* and y}\neq\text{*} \\ 500 & \text{otherwise} \end{cases} \quad (3)$$

Experiments setup

Five distinct experiments were performed to evaluate the standard GE and NS-GE with the four different descriptors using the BNF-Koza grammar and another five for BNF-ONeill. Each experiment consisted on 100 evolutionary runs. For each method, we have measured performance in terms of the success or hit rate (number of runs evolving individuals that eat all 89 food items) and the average quality of hits, (average number of steps necessary to eat all the 89 items). We also have measured the average number of food items eaten, the average number of generations for a hit, and finally the minimal number of steps of a hit. The tableau in Table 1 shows the settings and parameters of the experiments.

After some exploration, for almost all behavior descriptors we have chosen 10 for the parameter K, i.e. the number of nearest neighbors checked in NS (see Eq. 1). The only exception was the steps sequence descriptor: the best value for K was 5. We have used 50 as the maximum number of generations and a population of 500 individuals, which are widely used values in GP for the SFT. A Generation Gap of 90% means that in a population of 500 individuals the 50 best individuals will survive in the next generation and the worst 450 are replaced. Also, the standard genetic operators of mutation (point) and crossover (one-point) have been used.

Objective	Find an ant that follows food trails
Terminal Set	turn-left, turn-right, move, food-ahead
Behavior Descriptors	Amount of food eaten sampled at the end - (K=10, Euclidean), Amount of food eaten sampled 26 times - (K=10, Euclidean), Food eaten sequence - (K=10, Euclidean), Steps sequence - (K=5, A specific Euclidean distance adapted to the descriptor)
Grammars	BNF-O'Neill and BNF-Koza
Evolutionary Algorithm	Steady-State GA, Generation Gap: 0.9, Selection Mechanism: Roulette-Wheel
Initial Population	Randomly created with the following restrictions: Minimum Codons: 15 and Maximum Codons: 25
Parameters	Population Size: 500, Maximum Generations: 50 (without counting generation 0), Prob. Mutation: 0.01, Prob. Crossover: 0.9, Codon Size: 8, Wraps Limit: 10

Table 1: Summarizing Tableau for the SFT Problem.

Results

The experimental results for BNF-Koza and BNF-O'Neill can be seen in Table 2 and Table 3, respectively. They show that GE with NS, without any memory of past individuals, using any of the four behavioral descriptors, clearly outperforms standard fitness based GE in terms of hits rate for BNF-Koza, which is semantically equivalent to the original SFT problem. Standard GE for BNF-O'Neill, which defines a biased narrower search space, had a high hit percentage but was also outperformed by NS-GE for every behavior descriptor. NS-GE was also able to find more efficient programs (requiring less number of steps) than standard GE in both grammars. Anyhow, in spite that NS-GE with Steps Sequence descriptor found the most efficient solution, we cannot say that any of those descriptor is more efficient than the other: chance can play a role here, on average the number of required steps is very similar. Successful solutions evolved by NS-GE required a less number of evaluations, for both grammars (confirmed by the values on the Gen column in both tables). The average number of food items eaten is much higher in the NS-GE for every descriptor but that comes naturally from the higher success rate.

Even a simple descriptor that defines a very small behavior space (90 behaviors), and which is highly related with the fitness function: just counting the amount of food eaten in the end of the 650 steps, was able to attain a remarkable success rate. When we increase the number of samples (granularity of behavior characterization), we have increased the behavior space but we were also able to distinguish behaviors that were considered similar with just one sampling. Note that the amount of food eaten can never decrease and

BNF-Koza	Hits	Food	Gen	Steps	MinSteps
Fitness (GE)	8%	58.8	37.75	589.25	497
$Food_1$	42%	78.36	22.55	599.38	385
$Food_{26}$	48%	81.96	28.38	584.90	461
Food Seq.	33%	76.59	30.28	575.10	385
Steps Seq.	41%	77.88	23.78	587.92	331

BNF-O'Neill	Hits	Food	Gen	Steps	MinSteps
Fitness (GE)	77%	82.92	15.84	613.99	609
$Food_1$	95%	88.28	08.96	614.01	607
$Food_{26}$	100%	89.00	08.19	614.06	607
Food Seq.	94%	88.24	09.64	613.87	607
Steps Seq.	95%	88.28	08.96	614.01	607

Table 2: Results for BNF-Koza and BNF-O'Neill. For each behavior descriptor and objective based fitness, the columns give the percentage of hits, the average number of food items eaten, the average number of generations for a hit, the average and minimal number of steps for a hit.

26 samples imply sampling every 25 steps, which in combination with the form of the Santa Fe Trail impose strong constraints on the possible values of consecutive cells. The increase on the behavior description granularity has increased the performance in both grammars, but more experiments need to be made in order to assess if a too granular description is not necessary, distinguishing a lot of irrelevant behavior and slowing down evolution.

Considering all evolved programs in our experiments, the evolved code presented in Fig. 6 was able to complete the trail in the least number of steps: 331. This program outperforms the most efficient program known so far, evolved by the Constituent Grammatical Evolution algorithm (CGE) (Georgiou and Teahan, 2011). Note that CGE has attained a higher hit rate than NS-GE, but CGE is a much more complex algorithm than the standard GE, incorporating the concepts of constituent genes and conditional behavior switching, being able to decrease the actual search space, and explore programs in more useful areas. With a slight modification over the standard GE, using very intuitive behavior characterizations, by rewarding individuals that behave differently from the others, without any selective pressure to find the better ones, NS-GE has improved dramatically the performance on the very deceptive Santa Fe Trail Problem.

Conclusion and Future Work

Even though Grammatical Evolution shows competence on a series of problems where it has been applied, its effectiveness and efficiency on deceptive problems, such as the Santa Fe Trail, has been questioned (Robilliard et al., 2005). This is because problems of this type have some characteristics of real world problems, such as many local optima and large

```
ifelse food-ahead
  [ move move ]
  [ turn-left
    ifelse food-ahead
      [ move move ]
      [ turn-right ]
  ]
turn-right
ifelse food-ahead
  [ move ]
  [ turn-left ]
move
```

Figure 6: Best Program Evolved. Netlogo code of the best solution (331 steps) with the descriptor Steps Sequence.

search spaces, making them challenging and difficult for evolutionary algorithms to efficiently solve them (Langdon and Poli, 1998; O'Neill and Ryan, 2003). This work presented the first application of Novelty Search in Grammatical Evolution and benchmarked this approach in the Santa Fe Trail problem to investigate whether a search mechanism, which promotes behavioral novelty, would improve the performance of Grammatical Evolution in a deceptive problem. The results demonstrated a dramatic improvement in terms of both success rate and quality of solutions, which encourages further investigation and application to more problems such as Los Altos Hills and Maze Searching. Furthermore, more work is required for the investigation of how the application of Novelty Search in Grammatical Evolution affects the genotype and phenotype bloating, a known issue with the Santa Fe Trail problem.

Acknowledgments

This work was supported by FCT strategic project PEst−OE/EEI/UI0434/2011.

References

Banzhaf, W., Francone, F. D., and Nordin, P. (1996). The effect of extensive use of the mutation operator on generalization in genetic programming using sparse data sets. In *In Parallel Problem Solving from Nature IV, Proceedings of the International Conference on Evolutionary Computation, edited by*, pages 300–309. Springer Verlag.

Burke, E. K., Gustafson, S., Kendall, G., Krasnogor, N., and Gustafson, E. K. B. S. (2003). Is increased diversity in genetic programming beneficial? an analysis of lineage selection. In *Congress on Evolutionary Computation*, pages 1398–1405. IEEE Press.

Burke, E. K., Gustafson, S. M., Kendall, G., and Krasnogor, N. (2002). Advanced population diversity measures in genetic programming. In *Proceedings of the 7th International Conference on Parallel Problem Solving from Nature*, PPSN VII, pages 341–350, London, UK, UK. Springer-Verlag.

Ciesielski, V. and Mawhinney, D. (2002). Prevention of early convergence in genetic programming by replacement of similar

programs. In Fogel, D. B., El-Sharkawi, M. A., Yao, X., Greenwood, G., Iba, H., Marrow, P., and Shackleton, M., editors, *Proceedings of the 2002 Congress on Evolutionary Computation CEC2002*, pages 67–72. IEEE Press.

de Jong, E. D., Watson, R. A., and Pollack, J. B. (2001). Reducing bloat and promoting diversity using multi-objective methods. In Spector, L., Goodman, E. D., Wu, A., Langdon, W. B., Voigt, H.-M., Gen, M., Sen, S., Dorigo, M., Pezeshk, S., Garzon, M. H., and Burke, E., editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, pages 11–18, San Francisco, California, USA. Morgan Kaufmann.

Dempsey, I., O'Neill, M., and Brabazon, A. (2006). Adaptive trading with grammatical evolution. In *Proceedings of the 2006 IEEE Congress on Evolutionary Computation*, pages 9137–9142, Vancouver. IEEE Press.

Doncieux, S. and Mouret, J.-B. (2010). Behavioral diversity measures for evolutionary robotics. In *IEEE Congress on Evolutionary Computation*, pages 1–8. IEEE.

Doucette, J. and Heywood, M. I. (2010). Novelty-based fitness: an evaluation under the santa fe trail. In *Proceedings of the 13th European conference on Genetic Programming*, EuroGP'10, pages 50–61, Berlin, Heidelberg. Springer-Verlag.

Ekárt, A. and Németh, S. Z. (2002). Maintaining the diversity of genetic programs. In Foster, J. A., Lutton, E., Miller, J., Ryan, C., and Tettamanzi, A. G. B., editors, *Genetic Programming, Proceedings of the 5th European Conference, EuroGP 2002*, volume 2278 of *LNCS*, pages 162–171, Kinsale, Ireland. Springer-Verlag.

Georgiou, L. (2012). *Constituent Grammatical Evolution*. Phd thesis, School of Computer Science, Bangor University.

Georgiou, L. and Teahan, W. J. (2006). jge: a java implementation of grammatical evolution. In *Proceedings of the 10th WSEAS international conference on Systems*, ICS'06, pages 410–415, Stevens Point, Wisconsin, USA. World Scientific and Engineering Academy and Society (WSEAS).

Georgiou, L. and Teahan, W. J. (2010). Grammatical evolution and the santa fe trail problem. In Filipe, J. and Kacprzyk, J., editors, *IJCCI (ICEC)*, pages 10–19. SciTePress.

Georgiou, L. and Teahan, W. J. (2011). Constituent grammatical evolution. In Walsh, T., editor, *IJCAI*, pages 1261–1268. IJCAI/AAAI.

Gomes, J. C., Urbano, P., and Christensen, A. L. (2012). Introducing novelty search in evolutionary swarm robotics. In Dorigo, M., Birattari, M., Blum, C., Christensen, A. L., Engelbrecht, A. P., Groß, R., and Stützle, T., editors, *ANTS*, volume 7461 of *Lecture Notes in Computer Science*, pages 85–96. Springer.

Koza, J. R. (1991). Genetic evolution and co-evolution of computer programs. In Langton, C. T. C., Farmer, J. D., and Rasmussen, S., editors, *Artificial Life II*, volume X of *SFI Studies in the Sciences of Complexity*, pages 603–629. Addison-Wesley, Santa Fe Institute, New Mexico, USA.

Koza, J. R. (1992). *Genetic programming: on the programming of computers by means of natural selection*. MIT Press, Cambridge, MA, USA.

Langdon, W. B. and Poli, R. (1998). Why ants are hard. In Koza, J. R., Banzhaf, W., Chellapilla, K., Deb, K., Dorigo, M., Fogel, D. B., Garzon, M. H., Goldberg, D. E., Iba, H., and Riolo, R., editors, *Genetic Programming 1998: Proceedings of the Third Annual Conference*, pages 193–201, University of Wisconsin, Madison, Wisconsin, USA. Morgan Kaufmann.

Legg, S. and Hutter, M. (2005). Fitness uniform deletion: a simple way to preserve diversity. In Beyer, H.-G. and Reilly, U.-M. O., editors, *Genetic and Evolutionary Computation Conference, GECCO 2005, Proceedings, Washington DC, USA, June 25-29, 2005*, pages 1271–1278. ACM.

Lehman, J. and Stanley, K. O. (2008). Exploiting open-endedness to solve problems through the search for novelty. In *Proc. of the Eleventh Intl. Conf. on Artificial Life (ALIFE XI)*, Cambridge, MA. MIT Press.

Lehman, J. and Stanley, K. O. (2010). Efficiently evolving programs through the search for novelty. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2010)*. ACM.

Lehman, J. and Stanley, K. O. (2011). Abandoning objectives: Evolution through the search for novelty alone. *Evolutionary Computation*, 19(2):189–223.

Luke, S. (1998). Genetic programming produced competitive soccer softbot teams for robocup97. In *University of Wisconsin*, pages 214–222. Morgan Kaufmann.

Naredo, E., Trujillo, L., and Martinez, Y. (2013). Searching for novel classifiers. In Krawiec, K., Moraglio, A., Hu, T., Uyar, A. S., and Hu, B., editors, *Proceedings of the 16th European Conference on Genetic Programming, EuroGP 2013*, volume 7831 of *LNCS*, pages 145–156, Vienna, Austria. Springer Verlag.

O'Neill, M. and Ryan, C. (2001). Grammatical evolution. *IEEE Transactions on Evolutionary Computation*, 5(4):349–358.

O'Neill, M. and Ryan, C. (2003). *Grammatical Evolution: Evolutionary Automatic Programming in an Arbitrary Language*. Kluwer Academic Publishers, Norwell, MA, USA.

Robilliard, D., Mahler, S., Verhaghe, D., and Fonlupt, C. (2005). Santa fe trail hazards. In Talbi, E.-G., Liardet, P., Collet, P., Lutton, E., and Schoenauer, M., editors, *7th International Conference on Artificial Evolution EA 2005*, volume 3871 of *Lecture Notes in Computer Science*, pages 1–12, Lille, France. Springer. Revised Selected Papers.

Rosca, J. (1995). Entropy-driven adaptive representation. In *Proceedings of the Workshop on Genetic Programming: From Theory to Real-World Applications*, pages 23–32. Morgan Kaufmann.

Ryan, C., Collins, J., Collins, J., and O'Neill, M. (1998). Grammatical evolution: Evolving programs for an arbitrary language. In *Lecture Notes in Computer Science 1391, Proceedings of the First European Workshop on Genetic Programming*, pages 83–95. Springer-Verlag.

Yan, W. and Clack, C. D. (2009). Behavioural gp diversity for adaptive stock selection. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, GECCO '09, pages 1641–1648, New York, NY, USA. ACM.