Genetic Code Degeneracy: Implications for Grammatical Evolution and Beyond

Michael O'Neill & Conor Ryan

Dept. of Computer Science and Information Systems
University of Limerick
Ireland
{Michael.ONeill|Conor.Ryan}@ul.ie

Abstract. Grammatical Evolution (GE) is a grammar-based GA which generates computer programs. GE has the distinction that its input is a BNF, which permits it to generate programs in any language, of arbitrary complexity. Part of the power of GE is that it is closer to natural DNA than other Evolutionary Algorithms, and thus can benefit from natural phenomena such as a separation of search and solution spaces through a genotype to phenotype mapping, and a genetic code degeneracy which can give rise to silent mutations that have no effect on the phenotype. It has previously been shown how runs of GE are competitive with GP, and in this paper we analyse the feature of genetic code degeneracy, and its implications for genotypic diversity. Results show that genetic diversity is improved as a result of degeneracy in the genetic code for the problem domains addressed here.

1 Introduction

Grammatical Evolution (GE) is a grammar-based, variable length, linear genome system which is capable of generating programs or expressions in any language. Rather than the functions and terminals associated with GP [3], GE takes a BNF specification of a language, or subset thereof, from which it can subsequently generate compilable code. The BNF is used to build a program by applying production rules to elements of the non-terminal set of the BNF definition, in a mapping process to generate the output code from a simple binary string.

GE has been successfully applied to a number of diverse problem domains such as, symbolic regression [9], finding trigonometric identities [10], symbolic integration [10], and the Santa Fe trail [7]. The results compared favorably with systems such as GP, and has been shown to outperform GP [7].

In the spirit of Artificial Life, one definition of which is,

[the] field of study devoted to understanding life by attempting to abstract the fundamental dynamical principles underlying biological phenomenon, and recreating these dynamics in other physical media, such as computers, making them accessible to new kinds of experimental manipulation and testing [5], we have developed a system to generate programs which attempts to harness some of

the features of the genetic machinery of living organisms which are theorised to have an impact on the phenomenon of evolution [1]. This paper focuses on genetic code degeneracy, a characteristic of the genetic code in biological organisms which has been incorporated into GE, in an attempt to analyse the role this feature plays in maintaining genotypic diversity.

2 Grammatical Evolution

When tackling any problem with GE, a suitable BNF definition must first be decided upon. The BNF can be either the specification of an entire language, or perhaps more usefully, a subset of a language geared towards the problem at hand.

The genotype is then used to map the start symbol onto terminals by reading genes of 8 bits to generate a corresponding integer value, from which an appropriate production rule is selected. A rule is selected by the Integer Gene Value MOD the Number of Production Rules for the current non-terminal.

Considering the following rule,

```
(1) <code> :: =  <line>              (A)
                |<code><line>        (B)
```

i.e. given the non-terminal *code* there are two production rules to select from. If we assume the gene being read produces the integer 6, then $6\ MOD\ 2 = 0$ would select rule (A) $< line >$. Each time a production rule has to be selected to map from a non-terminal, another gene is read, and in this way, the system traverses the genome.

Given an 8 bit binary number, each gene can represent 256 distinct integer values. However, many of these integer values can represent the same production rule, taking production rule 5 as an example, if the current gene value was 6, then $6\ MOD\ 3 = 0$ would select rule (A) $left()$ as shown above. The same rule would be chosen if the gene value was 3, 9, 12, etc.

3 The Problem Spaces

Two problem domains were examined at which GE was previously found to be successful, namely a symbolic regression problem [9], and the Santa Fe ant trail [7]. For the Symbolic Regression problem the system is given a set of input and output pairs, and must determine the function that maps one onto the other. The particular function examined is $f(x) = X^4 + X^3 + X^2 + X$ with the input values in the range $[-1.. + 1]$. To determine the fitness of an individual program 20 test points were applied in the input range, and the fitness was taken as the sum of the error. The objective of the Santa Fe ant trail is to find a computer program to control an artificial ant so that it can find all 89 pieces of food located on a non-continuous trail within a specified number of time steps. The ant can only turn left, right, move forward one square. It may also look ahead one square in the direction it's facing to determine if that square contains a piece of food.

4 Results

Two sets of experiments were carried out for each problem domain. Fifty runs were carried out where genetic code degeneracy was removed so far as possible by reducing the number of bits in a gene to the lowest possible value that can still represent the maximum number of productions rules belonging to any one non-terminal. Another fifty runs were carried out where degeneracy was present as in the standard GE implementation using 8 bit genes.

Two measures were used to give an indication of genotypic diversity, the first is a measure which we have termed the *mean variety*. This measure was obtained by calculating the average of the variances at each bit locus on the genome.

With a population size of 500 this meant that the greater the variance in a population the closer the mean variety measure is to 0.25. The aim of this measure is to attempt to establish how different the individual genotypes in any given population are.

The other measure used was the number of unique individuals in a population, which can also be used to some extent to illustrate the genetic diversity within a population[4].

Results for both of these measures over both problem domains can be seen in Figures 1 2. As can be seen from these graphs the code degeneracy is having a marked effect on the mean variety measure, and on the number of unique individuals.

A measure of the average number of invalid individuals in each generation over the 50 runs was carried out for both problem domains, and results show that when the degeneracy is removed from the genetic code, the number of invalid individuals increases significantly over a run when compared to the case where a degenerate code exists.

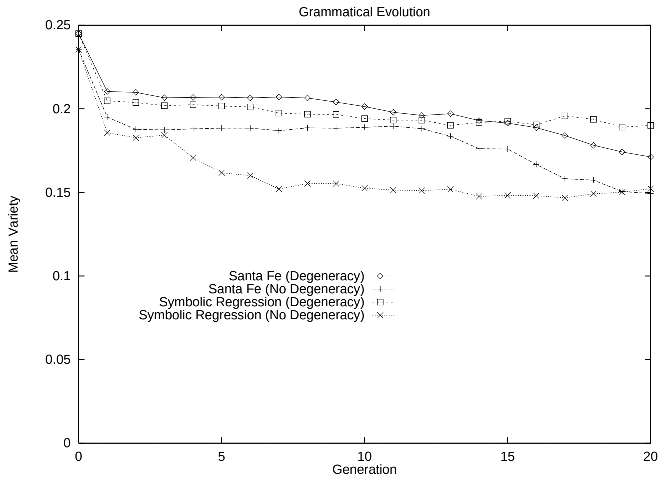

Fig. 1. Genetic Code Degeneracy and Mean Variety

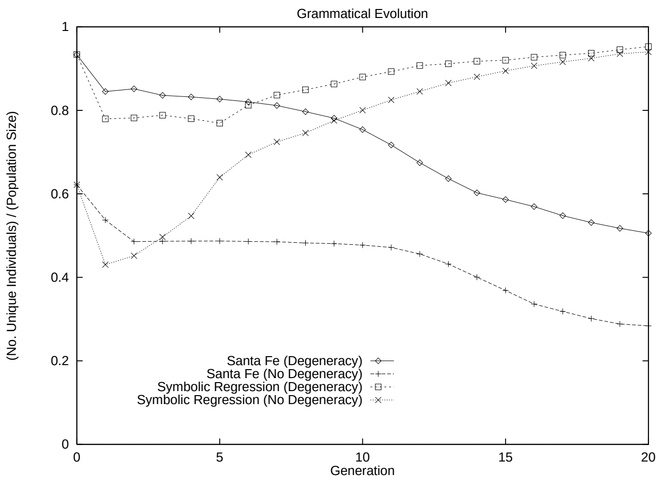

Fig. 2. Genetic Code Degeneracy and Unique Individuals

5 Discussion

In the standard GE implementation 8 bit genes are used, which can represent 256 distinct integer values. Therefore, in this state GE can represent 256 productions for each $non-terminal$ in the grammar. In the case of the Santa Fe trail the maximum number of productions that any one $non-terminal$ has is 3, and for symbolic regression problems this number is 4. As a result, the minimum number of bits any gene can have in the case of these problems is two, as this can represent a maximum of 4 distinct productions. It follows that in the case of the symbolic regression problem, all degeneracy has been removed, while it still exists to a very small extent in the Santa Fe trail problem.

Given the number of invalid individuals produced when genetic code degeneracy is removed, it is clear that degeneracy is responsible for the preservation of the functionality of the phenotype, while still allowing unrestricted search of the genotypic search space. Based on the two measures used to give an indication of the genotypic diversity in a population, the results clearly show that degeneracy in the genetic code is having a beneficial effect on genotypic diversity in the population.

Kimura's neutral theory of molecular evolution [2], suggests that molecular evolution is as a result largely of mutations that have no effect on the phenotype, i.e. functionality. Kimura mentions that this could also be responsible for maintaining genetic diversity within a population. The results produced by GE would reflect the basis for this theory within our artificial population, which is occurring due to the genetic code degeneracy which is part of the genotype to phenotype mapping process.

Following on from Kimura, others have noted the benefits that neutral evolution can bring to the evolutionary process [8] [6]. In particular in [6] it has been suggested that the maximum fitness attainable on a fitness landscape with a degree of neutrality increases as the degree of neutrality increases. Further investigation is required to determine the ramifications, if any, for GE based on these and other observations.

6 Conclusions

The results presented here show that genetic code degeneracy is having a clear beneficial effect on the genotypic diversity, and in the preservation of valid phenotypes, during runs of GE. The benefits of a complex mapping process show that the one-to-one genotype to phenotype mapping that is so prevalent in other evolutionary algorithms is not necessarily a good idea. We have previously shown that by harnessing features from the genetic system of biological organisms that the performance of GE can be enhanced, and as such there are clear advantages to be gained by using more biologically inspired techniques within evolutionary algorithms. It follows, that by modifying the current genotype to phenotype mapping process to one that is even closer to that of biological organisms, we may find that the performance of GE will benefit to an even greater extent.

Taking on board these findings, and the fact that GE has proven successful across diverse problem domains, it has been shown that GE is a powerful approach to automatically generating programs in any language.

References

1. Elseth Gerald D., Baumgardner Kandy D. 1995.
   Principles of Modern Genetics. *West Publishing Company.*
2. Kimura, M. 1983. The Neutral Theory of Molecular Evolution. Cambridge University Press.
3. Koza, J. 1992. *Genetic Programming.* MIT Press.
4. Langdon, W. 1998. Genetic Programming and Data Structures. Kluwer Academic Publishers.
5. Langton, C.G. 1992. Preface. In C.G. Langton, C.Taylor, J.D. Farmer, and S.Rasmussen, editors, Artificial Life II.
6. Newman M.E.J., Engelhardt Robin. Effects of neutral selection on the evolution of molecular species. In *Proc.R.Soc.London B*, **265** pages 1333-1338.
7. O'Neill M., Ryan C. 1999. Evolving Multi-line Compilable C Programs. In *Proceedings of the Second European Workshop on Genetic Programming 1999.*
8. Reidys C., Forst C.V., Schuster P. Replication and Mutation on Neutral Networks. Santa Fe Institute Working Paper 98-04-036.
9. Ryan C., Collins J.J., O'Neill M. 1998. Grammatical Evolution: Evolving Programs for an Arbitrary Language. In *Proceedings of the First European Workshop on Genetic Programming*, pages 83-95.
10. Ryan C., O'Neill M. Grammatical Evolution: A Steady State Approach. In *Late Breaking Papers, Genetic Programming 1998*, pages 180-185.