Exploring Grammatical Evolution

Robert G. Bradley

School of Computer Science & Informatics
University College Dublin

Abstract

This paper explores an area within Evolutionary Computation called Grammatical Evolution [8]. This approach is in contrast to Genetic Programming, where syntax trees are evolved. The approach taken in GE is based on ideas and processes in genetics. In nature there is a clear separation between the genotype (the genetic material DNA), and the phenotype (the actual living specimen). GE attempts to mimic this model. The equivalent genetic material in GE takes the form of variable length binary strings. These binary strings are composed of 8bit parts called codons. These codons are translated to integer equivalents, and it is these integers which are used to select production rules from a grammar. So the grammar sits between the genotype and phenotype and controls the mapping process. Grammatical Evolution can be seen as an exciting development in Evolutionary Computation. This model has been in existence in nature for millions of years and has proven itself to be extremely successful. In this paper we use GE to solve three benchmark problems. A population is evolved and their fitness value (a measure of their ability to solve the problem) is recorded and graphed. From these results we can see that GE has the ability to solve relatively complex problems, making it a welcome extension to the EC world.

1. Introduction

The idea of computers evolving solutions to problems is an extremely exciting one. Throughout history man has been learning from nature. We as a race are constantly evolving. This evolutionary process includes the developments in technology since the beginning of mankind. From the Stone Age club and ax to the modern day supercomputers, technology has been developing at an exponential rate. Evolutionary Computation can potentially be seen as a major factor in the future of technology. We are taking things to the next level when we start talking about technology which can evolve largely independent of human intervention. At first glance one may think this is a conceptual paper for a science fiction movie. Your ordinary Joe Bloggs on the street would probably laugh at the idea. The reality is somewhat different. The area of evolutionary computation is very active today. Computer Scientists are using it to develop a large range of solutions to problems in market trading, corporate prediction, circuit design, games AI, Bond rating, and automatic music generation. It has even reached the stage where these systems are coming up with novel solutions to problems and these solutions are being patented. Imagine – inventions by computers.

Both Genetic Algorithms and Genetic Programming come under the umbrella term Evolutionary Computation. Genetic Algorithms have been used to solve problems since the 1960s. This involves the evolution of binary strings. These binary strings are then decoded into a solution and this solution is evaluated and given a fitness value. The population then evolves based on a replacement strategy which looks at the fitness value of each solution in deciding on which solutions to keep, and which to destroy [1]. Genetic Programming can see as a similar process. However in GP the evolutionary processes of mutation and crossover are implemented on a syntax tree, which represents the solution. So mutation and crossover is acting on the phenotype rather than the genotype as seen in GA. Grammatical Evolution can be generally seen as a marriage of GA and GP. A GE system is composed of a search engine, an objective function and a grammar. The search engine is independent from the other components. We have an option to use a Genetic Algorithm for our search engine, however some other search strategy may be used. The binary strings evolved by the search engine are used to drive the selection of production rules from a grammar, for example we could use a BNF grammar. The output is a solution in a programming language like C++ for example. Here we notice that there is a definite divide between the phenotype and genotype [2], giving rise to a modular system with greater flexibility for future development. There have been many improvements to GE and this is an active and ongoing area of research.
Section 2 of this paper takes a look at Grammatical Evolution and the biology processes which inspired its development. This is followed by Section 3 which documents our experiments and results which involved applying GE to three problems: Cart Centering, Intertwined Spirals, and Santa Fe Ant Trail.

Finally Section 5 captures the conclusions for this paper.

2. A Look at Grammatical Evolution

Now let us take a look at GE in a little more detail. Ultimately the goal of a GE system is to find a solution to a problem. How can a GE possibly solve a problem? A GE system is made up of a search engine, an objective function, and a grammar. The search engine will initialize a population of variable length binary strings, just like in biology where we have variable length strings of DNA. Mutation and crossover are applied to these strings. It is this change in the binary strings which manifests itself as the search. The binary strings are composed of 8 bit sub strings called codons.

These binary strings are then read codon by codon. Each codon is translated to the integer representation. This integer is then use to select a production rule from a grammar. This grammar contains a collection of production rules, and it may also contain domain knowledge to bias the system in solving a particular problem. [3] The output is a program in an arbitrary language. We will end up with a collection of varying solution programs. How do we know which programs are close to solving the problem? How do we know which programs are useless? To answer these questions we pass each program to a fitness function. This fitness function evaluates the program and assigns it a fitness value based on how close it is to solving the problem.

Now that each solution has been assigned a fitness value we can let nature take its course through natural selection. As we are talking about computers we have control over this process. There are various replacement strategies in circulation including the steady-state replacement strategy where the parent with the lowest fitness is replaced by the solution with the best fitness from the next generation.

Once the replacement strategy has been applied to the solutions, the binary strings which created these programs are fed back into the search engine. This means the whole process is a cycle that will continue to cycle for n generations. As the GE process unfolds and the number of generations increases the average fitness of the population also increases. The idea is that each generation will be slightly better than the previous generation. However, it is not as simple as that. A population may converge after a certain number of generations.

3. Experiments

3.1 Setup

This section looks in detail at the three problems we tackled with Grammatical Evolution. The three problems were Cart Centering, Intertwined Spirals, and Santa Fe Ant Trail. The GE library we used was libGE [9] which is the standard GE library available for download online. A library called EO [10] was used for our search engine. EO is a comprehensive library for developing genetic algorithms. Here are the settings used for all experiments.

Number of runs: 30
Population size: 500
Number of generations: 20
Probability of crossover: 0.9
Probability of mutation: 1.0
Probability of bit-flip mutation: 0.01

3.2 Santa Fe Ant Trail

Here we set out to see if we could solve this benchmark problem using Grammatical Evolution. The Santa Fe Ant Trail problem is a well known problem which involves controlling an ant in a grid space so that it finds the food scattered around the grid [4]. There can be many local optima as there can be many locations of food within the grid, some being more substantial than others. The ant can turn left and right. It can also move one place forward, and in addition the ant may look into the square directly in front of it to determine whether or not it contains food.

The 3 charts below show the performance of GE on the Santa Fe Ant Trail problem.

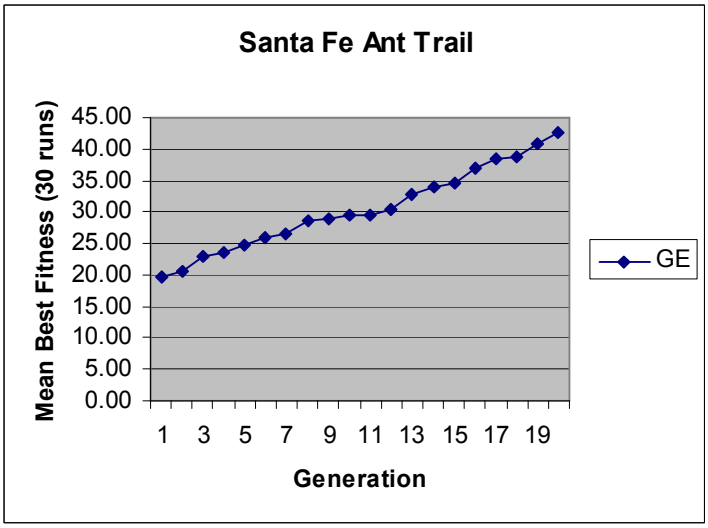

Figure 1.

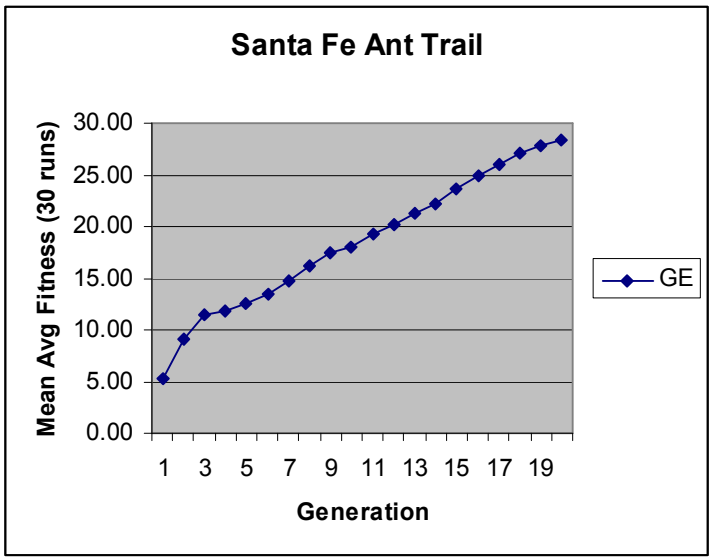

Figure 2.

3.3 Cart Centering

In this problem GE was given the task of solving the cart centering problem.

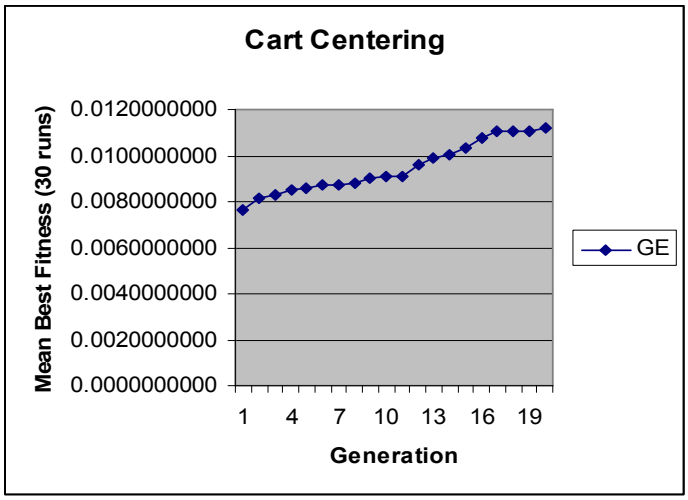

Figure 3.

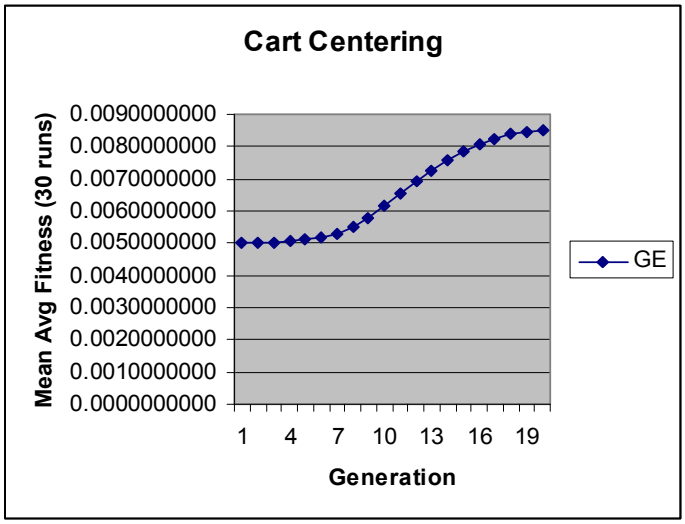

Figure 4.

3.4 Intertwined Spirals

Finally GE is applied to the intertwined spirals problem. Below are three charts showing GE performance. [7]

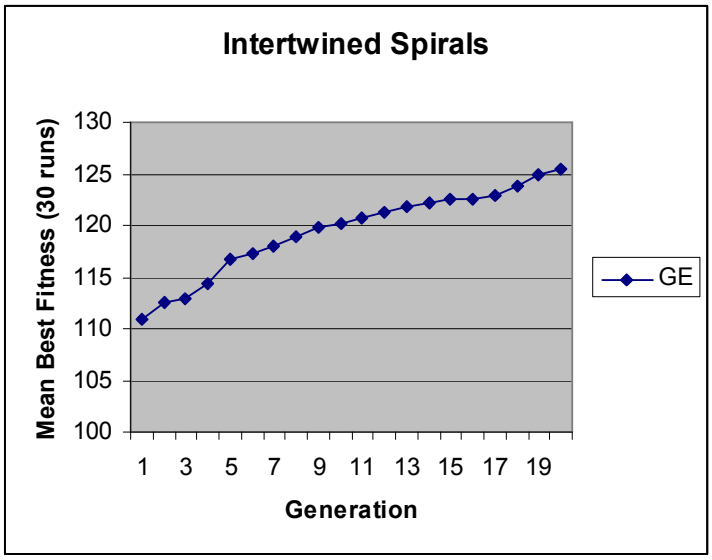

Figure 5.

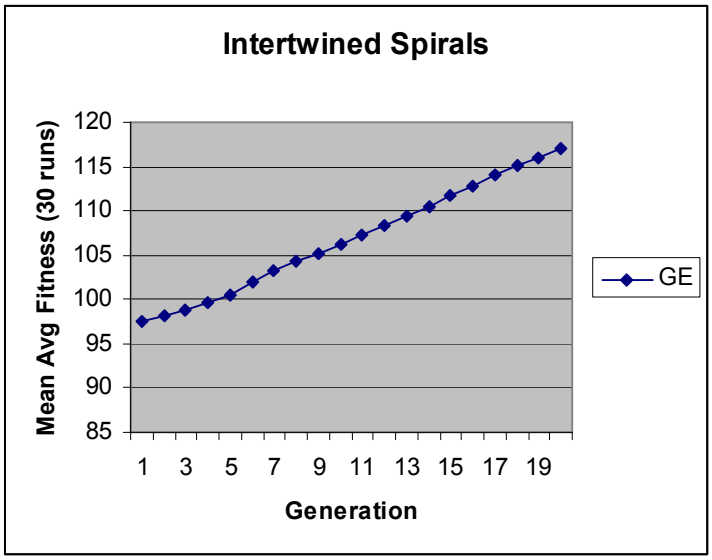

Figure 6.

4. Conclusions & Future Work

Having applied GE to three relatively difficult problems it is clearly evident from the results that GE has the ability to develop good solutions to the aforementioned problems. The fitness function measures each individuals ability to solve the problem for each generation. If you look at the graphs about you will notice that the fitness value steadily increases as the number of generations increase. This gradual improvement in fitness over time is exactly how nature behaves in the real world.

Although we have seen GE solve problems there is still considerable room for improvement. In the future I wish to further explore possible ways to improve GE´s performance. One development which has seen an increase in GE performance is piGE [2]. There are an infinite number of ways to tweak GE and it is a case of tweaking the system in the hope of improvement. Further research is need to explore the exciting world of GE today.

References

1. Brabazon A., O'Neill M. 2006. *Biologically Inspired Algorithms for Financial Modelling.* Springer.
2. Brabazon, A., O'Neill, M., Nicolau, M 2006 *piGrammatical Evolution*
3. O'Neill, M., Ryan, C. 2004 *Under the Hood of Grammatical Evolution*
4. Langdon, W.B., Poli, R. 1998 *Why Ants are Hard*
5. Muhammad Atif Azad, R., Ryan, C. *Structural Emergence with Order Independent Representations*
6. www.primordion.com/Xholon/samples/cartcentering_NoGP.html
7. Soule, Terence. *Cooperative Evolution on the Intertwined Spirals Problem*
8. O'Neill, M., Ryan, C. (2003). *Grammatical Evolution: Evolutionary Automatic Programming in an Arbitrary Language. Klewer Academic Publishers.*
9. libGE version 0.26 *http://bds.ul.ie/libGE/*
10. EO version 1 *https://sourceforge.net/projects/eodev/*