GEVA: Grammatical Evolution in Java*

Michael O’Neill†    Erik Hemberg    Conor Gilligan
Eliott Bartley    James McDermott
Anthony Brabazon

Natural Computing Research & Applications Group
Complex & Adaptive Systems Lab
University College Dublin
Ireland
m.oneill@ucd.ie

We are delighted to announce the release of GEVA [1] an open source software implementation of Grammatical Evolution (GE) in Java. Grammatical Evolution in Java (GEVA) was developed at UCD’s Natural Computing Research & Applications group (http://ncra.ucd.ie).

1 What is GE?

Grammatical Evolution (GE) (e.g., [2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]) is a grammar-based form of Genetic Programming [15]. It marries principles from molecular biology to the representational power of formal grammars. GE’s rich modularity gives a unique flexibility, making it possible to use alternative search strategies, whether evolutionary, or other heuristic, be it stochastic or deterministic, and to radically change its behaviour by merely changing the grammar supplied. As a grammar is used to describe the structures that are generated by GE, it is trivial to modify the output structures by simply editing the plain text grammar. This is one of the main advantages that makes the GE approach so attractive. The genotype-phenotype mapping also means that instead of operating exclusively on solution trees, as in standard GP, GE allows search operators to be performed on the genotype (e.g., integer or binary chromosomes), in addition to partially derived phenotypes, and the fully formed phenotypic derivation trees themselves.

*Published in SIGEVOlution Vol. 3 Issue 2 pp. 17-22
†Corresponding Author
Figure 1: A screenshot of the GEVA GUI. When opened for the first time it adopts the parameter settings for the pattern matching example problem where the goal is to rediscover the string “geva”.

2 Features

GEVA has been released under GNU GPL version 3, and uses Java 1.5.

As well as providing the characteristic genotype-phenotype mapper of GE, an evolutionary search engine and a GUI are also provided.

GEVA comes out-of-the-box with a number of demonstration problems that can be easily switched between from the GUI or command line. Sample problems include simple String Pattern Matching, an LSystem generator, the Paint problem, and a number of classic Genetic Programming problems such as an example of Symbolic Regression, the Santa Fe ant trail and Even Five Parity.

A screenshot of the default GUI screen showing settings for the pattern matching problem can be seen in Fig. 1. The goal of the problem is to rediscover the string “geva”. Simple graphing support is also provided, which allows the user to observe various attributes of the population live during the course of
Figure 2: An example graph produced while running GEVA on the pattern matching example problem. Displayed are the best fitness and average fitness with error bars. It is also possible to observe the average codon length of individuals, the average expressed codon length and the number of invalid individuals in the population.

A run. The resulting graphs can then be saved for later use. Attributes that can be plotted include the best fitness, the average fitness (with error bars), the number of invalid (incompletely mapped individuals), the average number of codons in each individual, and the average number of expressed codons in the population. For an example see Fig. 2.

A number of tutorials have been developed to help the novice user get up to speed: from running the software out-of-the-box, to using the command-line parameters, writing your own grammars and fitness functions, to developing your own search engine. These include a tutorial describing a bonus demo problem, Battleship, that the interested user can add to a GEVA installation.

3 Design Overview

GEVA takes advantage of GE’s modular structure as outlined in Fig. 3. This allows us to create a framework in which any search engine algorithm can be used to generate the genotypes (the GEChromosome class) that are used to direct the GE Mapper’s use of the Grammar during the development of the output solution. In recent years this approach has included the adoption of a Particle Swarm algorithm and Differential Evolution as alternative search engines resulting in Grammatical Swarm [14] and Grammatical Differential Evolution [16] variants. GEVA facilitates the adoption of alternative search engines through the provision of an Algorithm interface. This will work correctly as long as a GE Mapper object is provided with a legal GEChromosome object, so any alternative
algorithm must ensure to map its search engine’s individual representation to a GEChromosome to generate an output solution. In this first release a standard Genetic Algorithm engine is provided with plans to add alternative engines in future releases. The current version uses individuals with (32-bit) integer codon values and adopts a corresponding integer mutation operator.

Grammars are made available to GEVA through plain text files that adopt BNF notation. A simple parser is provided which handles standard BNF and can also recognise special symbols including \texttt{GE\_CODON\_VALUE}, which returns the current codon’s numerical value as a terminal symbol to the developing output phenotype sentence.

Simply by altering the contents of a BNF text file you can radically change the output generated by GEVA. A number of studies have illustrated this flexibility: for example, grammars have been used to represent a diverse array of structures including binary strings, code in various programming languages (e.g., C, Scheme, Slang, Postscript), music, financial trading rules, 3D surfaces, and even grammars themselves (examples include [18, 4, 19, 20, 21, 22, 23, 24]). A number of demonstration grammars are provided in the example problems and are available through the GUI and from the command line.

4 How to find out more

GEVA is available for download from the UCD NCRA group website (http://ncra.ucd.ie/geva) or http://www.grammatical-evolution.org. Included in the release are instruc-
Figure 4: GEVA includes an interactive form of GE as an example problem that allows users to generate interesting LSystems. A similar approach was used to evolve the logo for the Natural Computing Research & Applications Group at UCD (see Fig. 5).

Figure 5: The UCD NCRA group logo was evolved using Grammatical Evolution. See [17] for more details.
tions on how to run GEV A out-of-the-box, and more detailed tutorials for those who wish to modify the software for new purposes. We also welcome feedback on the software as we plan to actively maintain the code, releasing new versions as features are added. A GEVA Google group has been set up to facilitate communication amongst the GEVA community [25]. We hope that GEVA will be a useful resource for the EC community and beyond.

5 Acknowledgments

We would like to thank past and present members of the UCD Natural Computing Research & Applications group, especially Tiberiu Simu, Jonathan Huggson, and Jeff Wright for initial testing and some additions to the code. GEVA was beta-tested by the students of COMP30290 Natural Computing, an elective offered by UCD School of Computer Science & Informatics [27], from September to November 2007, and by Patrick Middleburgh who developed the seed of the LSystem interface as part of his Final Year Computer Science project at UCD. We would also like to thank Miguel Nicolau for many interesting discussions that helped inform the design of GEVA, in particular the genotype-phenotype mapper which draws upon the design adopted in Miguel’s libGE C++ library. This publication has emanated from research conducted with the financial support of Science Foundation Ireland under Grants No. 08/IN.1/I1868 and No. 06/RFP/CMS042, and UCD Research Seed Funding.

References


Email: geva@googlegroups.com
